Intercommunication
Intercommunication
Eighth edition (June 2002)

This edition applies to the following WebSphere MQ V5.3 products:

- WebSphere MQ for AIX®
- WebSphere MQ for HP-UX
- WebSphere MQ for iSeries™
- WebSphere MQ for Linux for Intel
- WebSphere MQ for Linux for zSeries™
- WebSphere MQ for Solaris
- WebSphere MQ for Windows®
- WebSphere MQ for z/OS™

Unless otherwise stated, the information also applies to these products:

- MQSeries® for AT&T GIS (NCR) UNIX® V2.2.1
- MQSeries for Compaq NonStop Kernel V5.1
- MQSeries for Compaq OpenVMS Alpha V5.1
- MQSeries for Compaq OpenVMS VAX V2.2.1
- MQSeries for Compaq Tru64 UNIX, V5.1
- MQSeries for OS/2® Warp V5.1
- MQSeries for SINIX and DC/OSx V2.2.1
- MQSeries for Sun Solaris, Intel Platform Edition, V5.1

© Copyright International Business Machines Corporation 1993, 2002. All rights reserved.
US Government Users Restricted Rights – Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.
Contents

Figures ......................... xiii

Tables .......................... xv

About this book ............... xvii
Who this book is for ........... xvii
What you need to know to understand this book xvii
How to use this book .......... xxi
Appearance of text in this book .... xix
Terms used in this book......... xix

Summary of changes ........... xxi
Changes for this edition (SC34–6059–00) xxi
Changes for the previous edition (SC33–1872–05) xxi

Part 1. Introduction ........... 1

Chapter 1. Concepts of intercommunication .......... 3
What is intercommunication? .... 3
   How does distributed queuing work? .... 3
Distributed queuing components .... 7
   Message channels. .... 8
   Message channel agents .... 10
   Transmission queues .... 10
   Channel initiators and listeners .... 10
   Channel-exit programs .... 12
Dead-letter queues ............. 13
Remote queue definitions ....... 14
How to get to the remote queue manager .... 14
   Multi-hopping .... 14
   Sharing channels .... 14
   Using different channels .... 15
   Using clustering .... 16

Chapter 2. Making your applications communicate .......... 17
How to send a message to another queue manager .... 17
   Defining the channels .... 18
   Defining the queues .... 19
   Sending the messages .... 20
   Starting the channel .... 20
Triggering channels .... 20
Safety of messages .... 22
   Fast, nonpersistent messages .... 22
   Undelivered messages .... 23

Chapter 3. More about intercommunication .......... 25
Addressing information .... 25
What are aliases? .... 25
   Queue name resolution .... 26
Queue manager alias definitions .... 26
Outbound messages - remapping the queue manager name .... 26
Outbound messages - altering or specifying the transmission queue .... 27
Inbound messages - determining the destination .... 27
Reply-to queue alias definitions .... 28
   What is a reply-to queue alias definition? .... 28
   Reply-to queue name .... 29
Networks .... 29
   Channel and transmission queue names .... 29
   Network planner .... 31

Part 2. How intercommunication works ............... 33

Chapter 4. WebSphere MQ distributed-messaging techniques .......... 35
Message flow control .... 35
   Queue names in transmission header .... 36
   How to create queue manager and reply-to aliases .... 36
Putting messages on remote queues .... 37
   More about name resolution .... 38
Choosing the transmission queue .... 39
Receiving messages .... 40
   Receiving alias queue manager names .... 40
Passing messages through your system .... 41
   Method 1: Using the incoming location name .... 41
   Method 2: Using an alias for the queue manager .... 42
   Method 3: Selecting a transmission queue .... 42
   Using these methods .... 42
Separating message flows .... 42
Concentrating messages to diverse locations .... 44
Diverting message flows to another destination .... 45
Sending messages to a distribution list .... 46
   Reply-to queue .... 47
      Reply-to queue alias example .... 48
      How the example works .... 50
      How the queue manager makes use of the reply-to queue alias .... 51
      Reply-to queue alias walk-through .... 51
Networking considerations .... 52
Return routing .... 53
Managing queue name translations .... 53
Channel message sequence numbering .... 54
   Sequential retrieval of messages .... 55
   Sequence of retrieval of fast, nonpersistent messages .... 55
Loopback testing .... 56

Chapter 5. DQM implementation .......... 57
Functions of DQM .... 57
Message sending and receiving .... 58
   Channel parameters .... 59
Channel status and sequence numbers ........................................... 59
Channel control function ............................................................. 59
Preparing channels ........................................................................ 60
Channel states .................................................................................. 61
Adopting an MCA ............................................................................. 67
Stopping and quiescing channels .................................................... 67
Restarting stopped channels ............................................................ 69
In-doubt channels ............................................................................ 70
Problem determination .................................................................... 71

What happens when a message cannot be delivered? ....................... 71
Initialization and configuration files .................................................. 73
z/OS without CICS ........................................................................... 73
z/OS using CICS ............................................................................. 73
Windows systems ............................................................................ 73
OS/2, Compaq OpenVMS Alpha, Compaq NonStop Kernel, iSeries, and UNIX systems ......................................................... 74
VSE/ESA ......................................................................................... 75
Data conversion ................................................................................. 75
Writing your own message channel agents ..................................... 75

Chapter 6. Channel attributes ............................................................ 77
Channel attributes and channel types .............................................. 77
Channel attributes in alphabetical order ........................................... 79
Alter date (ALTDATE) ..................................................................... 80
Alter time (ALTTIME) ..................................................................... 80
Auto start (AUTOSTART) ................................................................. 80
Batch Heartbeat Interval (BATCHHB) ............................................. 80
Batch interval (BATCHINT) ............................................................... 81
Batch size (BATCHSZ) ..................................................................... 81
Channel name (CHANNEL) ............................................................... 82
Channel type (CHLTYPE) ................................................................. 83
CICS profile name ........................................................................... 83
Cluster (CLUSTER) .......................................................................... 83
Cluster namelist (CLUSNL) .............................................................. 83
Connection name (CONNNAME) ..................................................... 84
Convert message (CONVERT) .......................................................... 85
Description (DESCR) ..................................................................... 86
Disconnect interval (DISCINT) ......................................................... 86
Heartbeat interval (HBINT) ............................................................... 87
KeepAlive Interval (KAINT) .............................................................. 87
Local Address (LOCLADDR) ............................................................. 88
Long retry count (LONGRTY) ........................................................... 88
Long retry interval (LONGTMR) ....................................................... 89
LU 6.2 mode name (MODENAME) ................................................... 89
LU 6.2 transaction program name (TPNAME) ............................... 89
Maximum message length (MAXMSGL) ......................................... 90
Maximum transmission size ............................................................. 90
Message channel agent name (MCANAME) ..................................... 91
Message channel agent type (MCATYPE) ....................................... 91
Message channel agent user identifier (MCAUSER) ......................... 91
Message exit name (MSGEXIT) ......................................................... 92
Message exit user data (MSGDATA) ............................................... 92
Message-retry exit name (MREXIT) ................................................ 92
Message-retry exit user data (MREDATA) .................................... 92
Message retry count (MRRTY) ........................................................ 92
Message retry interval (MRTMR) ..................................................... 93
Network-connection priority (NETPRTY) ........................................ 93
Nonpersistent message speed (NPMSPEED) ................................... 93
Password (PASSWORD) ................................................................... 93
PUT authority (PUTAUT) ................................................................. 94
Queue manager name (QNAME) ...................................................... 95
Receive exit name (RCVEXIT) ........................................................ 95
Receive user data (RCVDATA) ....................................................... 95
Security exit name (SCYEXIT) ....................................................... 96
Security user data (SCYDATA) ....................................................... 96
Send exit name (SENDEXIT) .......................................................... 96
Send user data (SENDDATA) ........................................................ 96
Sequence number wrap (SEQWRAP) ............................................. 96
Sequential delivery ......................................................................... 97
Short retry count (SHORTRTY) ...................................................... 97
Short retry interval (SHORTTMR) .................................................. 97
SSL Cipher Specification (SSLCIPH) ............................................ 98
SSL Client Authentication (SSLCAUTH) ...................................... 98
SSL Peer (SSLPEER) ................................................................. 98
Target system identifier ................................................................. 99
Transaction identifier ..................................................................... 99
Transmission queue name (XMITQ) ............................................... 99
Transport type (TRPTYPE) ............................................................. 100
User ID (USERID) ..................................................................... 100

Chapter 7. Example configuration chapters in this book .................. 101
Network infrastructure ................................................................. 102
Communications software ............................................................ 102
How to use the communication examples .................................... 103
IT responsibilities .......................................................................... 104

Part 3. DQM in WebSphere MQ for UNIX systems, and Windows systems, and MQSeries for Compaq OpenVMS Alpha, Compaq NonStop Kernel, and OS/2 Warp .................................................. 105

Chapter 8. Monitoring and controlling channels on distributed platforms ................................................................. 109
The DQM channel control function ................................................. 109
Functions available ....................................................................... 110
Getting started with objects .......................................................... 112
Creating objects .......................................................................... 112
Creating default objects ............................................................... 113
Creating a channel ....................................................................... 113
Displaying a channel .................................................................. 114
Displaying channel status ............................................................. 114
Starting a channel ....................................................................... 115
Renaming a channel .................................................................... 115
Channel attributes and channel types ............................................ 115
Channel functions ........................................................................... 115
Create ......................................................................................... 115
Change ....................................................................................... 116
Delete .......................................................................................... 116
Display ....................................................................................... 116
Display Status ............................................................................. 116
Ping ............................................................................................ 116
Start ............................................................................................ 117
Stop ............................................................................................. 118
Reset .......................................................................................... 119
Resolve ........................................................................................ 119
<table>
<thead>
<tr>
<th>Chapter 15. Example configuration - IBM MQSeries for Compaq Tru64 UNIX</th>
<th>215</th>
</tr>
</thead>
<tbody>
<tr>
<td>Establishing a TCP connection</td>
<td>215</td>
</tr>
<tr>
<td>What next?</td>
<td>215</td>
</tr>
<tr>
<td>MQSeries for Compaq Tru64 UNIX configuration</td>
<td>215</td>
</tr>
<tr>
<td>Basic configuration</td>
<td>216</td>
</tr>
<tr>
<td>Channel configuration</td>
<td>216</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Chapter 16. Example configuration - IBM WebSphere MQ for HP-UX</th>
<th>219</th>
</tr>
</thead>
<tbody>
<tr>
<td>Configuration parameters for an LU 6.2 connection</td>
<td>219</td>
</tr>
<tr>
<td>Configuration worksheet</td>
<td>219</td>
</tr>
<tr>
<td>Explanation of terms</td>
<td>222</td>
</tr>
<tr>
<td>Establishing a session using HP SNAplus2</td>
<td>223</td>
</tr>
<tr>
<td>SNAplus2 configuration</td>
<td>223</td>
</tr>
<tr>
<td>APPC configuration</td>
<td>227</td>
</tr>
<tr>
<td>HP-UX operation</td>
<td>237</td>
</tr>
<tr>
<td>What next?</td>
<td>237</td>
</tr>
<tr>
<td>Establishing a TCP connection</td>
<td>237</td>
</tr>
<tr>
<td>What next?</td>
<td>238</td>
</tr>
<tr>
<td>WebSphere MQ for HP-UX configuration</td>
<td>238</td>
</tr>
<tr>
<td>Basic configuration</td>
<td>238</td>
</tr>
<tr>
<td>Channel configuration</td>
<td>238</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Chapter 17. Example configuration - IBM MQSeries for AT&amp;T GIS UNIX, V2.2</th>
<th>243</th>
</tr>
</thead>
<tbody>
<tr>
<td>Configuration parameters for an LU 6.2 connection</td>
<td>243</td>
</tr>
<tr>
<td>Configuration worksheet</td>
<td>243</td>
</tr>
<tr>
<td>Explanation of terms</td>
<td>246</td>
</tr>
<tr>
<td>Establishing a connection using AT&amp;T GIS SNA Server</td>
<td>246</td>
</tr>
<tr>
<td>Defining local node characteristics</td>
<td>247</td>
</tr>
<tr>
<td>Connecting to a partner node</td>
<td>249</td>
</tr>
<tr>
<td>Configuring a remote node</td>
<td>249</td>
</tr>
<tr>
<td>What next?</td>
<td>251</td>
</tr>
<tr>
<td>Establishing a TCP connection</td>
<td>251</td>
</tr>
<tr>
<td>What next?</td>
<td>251</td>
</tr>
<tr>
<td>MQSeries for AT&amp;T GIS UNIX configuration</td>
<td>252</td>
</tr>
<tr>
<td>Basic configuration</td>
<td>252</td>
</tr>
<tr>
<td>Channel configuration</td>
<td>252</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Chapter 18. Example configuration - IBM WebSphere MQ for Solaris.</th>
<th>257</th>
</tr>
</thead>
<tbody>
<tr>
<td>Loading the WebSphere MQ library to support SNA</td>
<td>257</td>
</tr>
<tr>
<td>Configuration parameters for an LU 6.2 connection using SunLink Version 9.1</td>
<td>258</td>
</tr>
<tr>
<td>Configuration worksheet</td>
<td>258</td>
</tr>
<tr>
<td>Explanation of terms</td>
<td>261</td>
</tr>
<tr>
<td>Establishing a connection using SunLink Version 9.1</td>
<td>262</td>
</tr>
<tr>
<td>SunLink 9.1 base configuration</td>
<td>262</td>
</tr>
<tr>
<td>Configuring a PU 2.1 server</td>
<td>262</td>
</tr>
<tr>
<td>Adding a LAN connection</td>
<td>263</td>
</tr>
<tr>
<td>Configuring a connection to a remote PU</td>
<td>265</td>
</tr>
<tr>
<td>Configuring an independent LU</td>
<td>266</td>
</tr>
<tr>
<td>Configuring a partner LU</td>
<td>268</td>
</tr>
<tr>
<td>Configuring the session mode</td>
<td>269</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Chapter 19. Setting up communication in Compaq OpenVMS Alpha systems</th>
<th>295</th>
</tr>
</thead>
<tbody>
<tr>
<td>Deciding on a connection</td>
<td>295</td>
</tr>
<tr>
<td>Defining a TCP connection</td>
<td>296</td>
</tr>
<tr>
<td>Sending end.</td>
<td>296</td>
</tr>
<tr>
<td>Receiving channels using Compaq (DIGITAL) TCP/IP services (UCX) for OpenVMS</td>
<td>296</td>
</tr>
<tr>
<td>Receiving channels using Cisco MultiNet for OpenVMS</td>
<td>297</td>
</tr>
<tr>
<td>Receiving channels using Attachmate PathWay for OpenVMS</td>
<td>298</td>
</tr>
<tr>
<td>Receiving channels using Process Software Corporation TCPware</td>
<td>298</td>
</tr>
<tr>
<td>Defining an LU 6.2 connection</td>
<td>299</td>
</tr>
<tr>
<td>SNA configuration.</td>
<td>299</td>
</tr>
<tr>
<td>Specifying SNA configuration parameters to WebSphere MQ.</td>
<td>301</td>
</tr>
<tr>
<td>Sample WebSphere MQ configuration</td>
<td>302</td>
</tr>
<tr>
<td>Problem solving</td>
<td>303</td>
</tr>
<tr>
<td>Defining a DECnet Phase IV connection</td>
<td>304</td>
</tr>
<tr>
<td>Sending end.</td>
<td>304</td>
</tr>
<tr>
<td>Receiving on DECnet Phase IV</td>
<td>304</td>
</tr>
<tr>
<td>Defining a DECnet Phase V connection.</td>
<td>304</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Chapter 20. Setting up communication in Compaq NonStop Kernel</th>
<th>307</th>
</tr>
</thead>
<tbody>
<tr>
<td>Deciding on a connection</td>
<td>307</td>
</tr>
<tr>
<td>SNA channels</td>
<td>307</td>
</tr>
<tr>
<td>LU 6.2 responder processes</td>
<td>309</td>
</tr>
<tr>
<td>TCP channels</td>
<td>309</td>
</tr>
<tr>
<td>Communications examples</td>
<td>309</td>
</tr>
<tr>
<td>SNAX communications example</td>
<td>317</td>
</tr>
<tr>
<td>ICE communications example</td>
<td>321</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Chapter 21. Message channel planning example for distributed platforms</th>
<th>323</th>
</tr>
</thead>
<tbody>
<tr>
<td>What the example shows</td>
<td>323</td>
</tr>
<tr>
<td>Queue manager QM1 example</td>
<td>325</td>
</tr>
<tr>
<td>Queue manager QM2 example</td>
<td>326</td>
</tr>
</tbody>
</table>

vi WebSphere MQ Intercommunication
What the example shows .............. 541
Queue manager QM1 example ........ 544
Queue manager QM2 example ........ 545
Running the example ................ 545
Expanding this example ............... 545

Part 6. DQM in MQSeries for VSE/ESA .... 547
Chapter 44. Example configuration - MQSeries for VSE/ESA .... 549
Configuration parameters for an LU 6.2 connection 549
  Configuration worksheet ............. 549
  Explanation of terms ................ 551
Establishing an LU 6.2 connection .... 552
  Defining a connection ............... 552
  Defining a session .................. 552
  Installing the new group definition 553
  What next? .......................... 553
Establishing a TCP connection ....... 554
MQSeries for VSE/ESA configuration 554
  Configuring channels ................ 554
  Defining a local queue ............. 557
  Defining a remote queue ............ 559
  Defining a SNA LU 6.2 sender channel 561
  Defining a SNA LU6.2 receiver channel 562
  Defining a TCP/IP sender channel .... 564
  Defining a TCP receiver channel .... 565

Part 7. Further intercommunication considerations .......... 567
Chapter 45. Channel-exit programs 571
What are channel-exit programs? ........ 571
  Processing overview .................. 572
  Channel security exit programs ....... 574
  Channel send and receive exit programs 578
  Channel send exit programs — reserving space 581
  Channel message exit programs ....... 581
  Channel message retry exit program .... 583
  Channel auto-definition exit program 583
  Transport-retry exit program ......... 584
Writing and compiling channel-exit programs 585
  WebSphere MQ for z/OS without CICS 587
  WebSphere MQ for z/OS using CICS 588
  WebSphere MQ for iSeries .......... 589
  MQSeries for OS/2 Warp .......... 589
  Windows 3.1 client ................. 591
WebSphere MQ for Windows server, WebSphere MQ client for Windows ........ 591
WebSphere MQ for AIX .......... 593
MQSeries for Compaq OpenVMS Alpha 595
MQSeries for Compaq Tru64 UNIX .... 596
WebSphere MQ for HP-UX .......... 597
MQSeries for AT&T GIS UNIX .... 598
WebSphere MQ for Solaris .......... 599
MQSeries for SINIX and DC/OSx .... 600
MQSeries for Compaq NonStop Kernel .... 600
Supplied channel-exit programs using DCE security services ............... 604
  What do the DCE channel-exit programs do? 604
  How do the DCE channel-exit programs work? 605
  How to use the DCE channel-exit programs 607
  SSPI security exit ................. 610

Chapter 46. Channel-exit calls and data structures ........ 611
Data definition files ................. 612
MQ_CHANNEL_EXIT – Channel exit ........ 614
  Syntax .................................. 614
  Parameters ................................ 614
  Usage notes ................................ 616
  C invocation ................................ 617
  COBOL invocation .......................... 617
  PL/I invocation .............................. 617
  RPG invocation (ILE) ....................... 617
  RPG invocation (OPM) ...................... 618
  System/390 assembler invocation ........ 618
MQ_CHANNEL_AUTO_DEF_EXIT – Channel auto-definition exit .......... 619
  Syntax .................................. 619
  Parameters ................................ 619
  Usage notes ................................ 619
  C invocation ................................ 620
  COBOL invocation .......................... 620
  RPG invocation (ILE) ....................... 620
  RPG invocation (OPM) ...................... 620
  System/390 assembler invocation ........ 620
MQ_TRANSPORT_EXIT – Transport retry exit .......... 622
  Syntax .................................. 622
  Parameters ................................ 622
  Usage notes ................................ 622
  C invocation ................................ 622
MQXWAIT – Wait in exit .................. 624
  Syntax .................................. 624
  Parameters ................................ 624
  C invocation ................................ 625
System/390 assembler invocation .......... 625
MQCD – Channel definition ............. 626
  Exit name fields ......................... 628
  Fields ..................................... 628
  C declaration ................................ 652
  COBOL declaration .......................... 654
  PL/I declaration .............................. 656
  RPG declaration (ILE) ....................... 658
  RPG declaration (OPM) ...................... 660
  System/390 assembler declaration ........ 662
  Visual Basic declaration .................. 664
MQCXP – Channel exit parameter .......... 666
  Fields ..................................... 666
  C declaration ................................ 677
  COBOL declaration .......................... 677
  PL/I declaration .............................. 678
  RPG declaration (ILE) ....................... 678
  RPG declaration (OPM) ...................... 679
  System/390 assembler declaration ........ 679
MQTXP – Transport exit parameter ........ 681
  Fields ..................................... 681
  C declaration ................................ 684
MQXWD – Exit wait descriptor 685
Fields 685
C declaration 686
System/390 assembler declaration 686

Chapter 47. Problem determination in DQM 687
Error message from channel control 687
Ping 687
Dead-letter queue considerations 688
Validation checks 688
In-doubt relationship 689
Channel startup negotiation errors 689
When a channel refuses to run 689

Retrying the link 692
Retry considerations 692
Data structures 692
User exit problems 692
Disaster recovery 693
Channel switching 693
Connection switching 694
Client problems 694
Terminating clients 694
Error logs 694

Error logs for OS/2, Windows and Windows 98 client 694
Error logs on UNIX systems 695
Error logs on DOS, Windows 3.1, and Windows 95 clients 695
Error logs on z/OS 695
Error logs on MQSeries for VSE/ESA 695
Error logs on MQSeries for Compaq NonStop Kernel 696

Part 8. Appendixes 697

Appendix A. Channel planning form 699
How to use the form 699

Appendix B. Constants for channels and exits 703
List of constants 703
MQ_* (Lengths of character string and byte fields) 703
MQCC_* (Completion code) 704
MQCD_* (Channel definition structure length) 704
MQCD_* (Channel definition structure version) 704
MQCDC_* (Channel data conversion) 704
MQCF_* (Channel capability flags) 704
MQCHT_* (Channel type) 705
MQCXP_* (Channel-exit parameter structure identifier) 705
MQCXP_* (Channel-exit parameter structure version) 705
MQFB_* (Feedback) 705
MQKAL_* (Keepalive interval) 705
MQMCAT_* (MCA type) 705
MQNPMS_* (Nonpersistent message speed) 706
MQPA_* (Put authority) 706
MQRC_* (Reason code) 706
MQSCA_* (SSL client authentication) 706
MQSID_* (Security identifier) 706
MQSIDT_* (Security identifier type) 707
MQTXP_* (Transport retry exit structure identifier) 707
MQTXP_* (Transport retry exit structure version) 707
MQXCC_* (Exit response) 707
MQXFT_* (Transmission protocol type) 707
MQXR_* (Exit reason) 708
MQXR2_* (Secondary exit response) 708
MQXT_* (Exit identifier) 708
MQXUA_* (Exit user area) 709
MQXWD_* (Exit wait descriptor structure identifier) 709
MQXWD_* (Exit wait descriptor version) 709

Appendix C. Queue name resolution 711
What is queue name resolution? 713
How queue name resolution works 714

Appendix D. Configuration file stanzas for distributed queuing 715

Appendix E. Notices 719
Trademarks 721

Index 723

Sending your comments to IBM 739
### Figures

1. Overview of the components of distributed queuing .................................. 4
2. Sending messages ................................................................................. 5
3. Sending messages in both directions ......................................................... 6
4. A cluster of queue managers .................................................................. 7
5. A sender-receiver channel ..................................................................... 8
6. A requester-server channel ................................................................... 9
7. A requester-sender channel .................................................................. 9
8. A cluster-sender channel ...................................................................... 10
9. Channel initiators and listeners ............................................................... 11
10. Sequence in which channel exit programs are called ........................... 13
11. Passing through intermediate queue managers ..................................... 14
12. Sharing a transmission queue ................................................................ 15
13. Using multiple channels ...................................................................... 15
14. The concepts of triggering .................................................................. 21
15. Queue manager alias ........................................................................... 27
16. Reply-to queue alias used for changing reply location ....................... 28
17. Network diagram showing all channels .................................................. 30
18. Network diagram showing QM-concentrators ...................................... 32
19. A remote queue definition is used to resolve a queue name to a transmission queue to an adjacent queue manager .................. 38
20. The remote queue definition allows a different transmission queue to be used ........................................................... 39
21. Receiving messages directly, and resolving alias queue manager name .... 40
22. Three methods of passing messages through your system ...................... 41
23. Separating messages flows ................................................................... 43
24. Combining message flows on to a channel .......................................... 44
25. Diverting message streams to another destination .................................. 45
26. Reply-to queue name substitution during PUT call ............................. 47
27. Reply-to queue alias example ............................................................... 49
28. Distributed queue management model .................................................. 58
29. Channel states ..................................................................................... 62
30. Flows between channel states ............................................................... 63
31. What happens when a message cannot be delivered ............................ 72
32. WebSphere MQ channel to be set up in the example configuration chapters in this book ................................. 101
33. Local LU window ................................................................................ 206
34. Mode window ...................................................................................... 207
35. CPI-C side information file for SunLink Version 9.0 ......................... 272
36. The message channel example for OS/2, Windows NT, and UNIX systems .......................................................... 324
37. The operations and controls initial panel ................................................. 342
38. Listing channels. .................................................................................. 343
39. Starting a system function .................................................................... 347
40. Stopping a function control .................................................................. 348
41. Starting a channel ................................................................................ 350
42. Testing a channel ................................................................................ 351
43. Stopping a channel .............................................................................. 353
44. Listing channel connections .................................................................. 355
45. Listing cluster channels ....................................................................... 356
46. Channel Initiator APPL definition ....................................................... 369
47. Channel Initiator initialization parameters .......................................... 370
48. Channel Initiator initialization parameters .......................................... 371
49. The first example for WebSphere MQ for z/OS .................................... 377
50. Sample configuration of channel control and MCA .............................. 384
51. The Message Channel List panel .......................................................... 385
52. The Message Channel List panel pull-down menus ............................ 387
53. The Channel pull-down menu .............................................................. 389
54. Sender/server Stop action window ....................................................... 392
55. Requester/receiver Stop action window ............................................... 393
56. The Reset Channel Sequence Number action window ....................... 395
57. The Resolve Channel action window .................................................... 396
58. An example of a sender channel Display Channel Status window ........ 397
59. An example of a receiver channel Display Channel Status window .... 397
60. The Ping action window ....................................................................... 398
61. The Exit confirmation secondary window ............................................. 399
62. The Copy action window ..................................................................... 400
63. The Create action window .................................................................. 401
64. Example of default values during Create for a channel ....................... 401
65. The Delete action window .................................................................... 402
66. The Find a Channel action window ...................................................... 402
67. The Include search criteria action window .......................................... 403
68. The Help pull-down menu ................................................................... 404
69. The Help choice pull-down menu ........................................................ 405
70. The sender channel settings panel ....................................................... 408
71. The sender channel settings panel - screen 2 ....................................... 408
72. The receiver channel settings panel ....................................................... 409
73. The receiver channel settings panel - screen 2 ....................................... 409
74. The server channel settings panel ........................................................ 410
75. The server channel settings panel - screen 2 ........................................ 410
76. The requester channel settings panel .................................................... 411
77. The requester channel settings panel - screen 2 .................................... 411
78. CICS LU 6.2 connection definition ....................................................... 418
79. Connecting two queue managers in WebSphere MQ for z/OS using CICS ................................................................. 433
80. Sender settings (1) .............................................................................. 434
81. Sender settings (2) .............................................................................. 435
82. Connection definition (1) ..................................................................... 435
83. Connection definition (2) ..................................................................... 436
84. Connection definition (1) ..................................................................... 436
85. Connection definition (2) ..................................................................... 437
86. Receiver channel settings (1) ............................................................... 437
87. Receiver channel settings (2) ............................................................... 438
88. Channel Initiator initialization parameters .......................................... 452
89. Channel Initiator initialization parameters .......................................... 453
<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Example of channel names</td>
<td>30</td>
</tr>
<tr>
<td>2</td>
<td>Three ways of using the remote queue definition object</td>
<td>37</td>
</tr>
<tr>
<td>3</td>
<td>Reply-to queue alias</td>
<td>51</td>
</tr>
<tr>
<td>4</td>
<td>Queue name resolution at queue manager QMA</td>
<td>54</td>
</tr>
<tr>
<td>5</td>
<td>Queue name resolution at queue manager QMB</td>
<td>54</td>
</tr>
<tr>
<td>6</td>
<td>Reply-to queue name translation at queue manager QMA</td>
<td>54</td>
</tr>
<tr>
<td>7</td>
<td>Channel attributes for the channel types</td>
<td>77</td>
</tr>
<tr>
<td>8</td>
<td>Negotiated HBINT value and the corresponding KAINT value</td>
<td>87</td>
</tr>
<tr>
<td>9</td>
<td>Functions available in OS/2, Windows systems, Compaq OpenVMS Alpha, Compaq NonStop Kernel, and UNIX systems</td>
<td>110</td>
</tr>
<tr>
<td>10</td>
<td>Channel programs for OS/2 and Windows systems</td>
<td>123</td>
</tr>
<tr>
<td>11</td>
<td>Channel programs for UNIX systems, Compaq OpenVMS Alpha, and Compaq NonStop Kernel</td>
<td>123</td>
</tr>
<tr>
<td>12</td>
<td>Default outstanding connection requests on OS/2 and Windows</td>
<td>131</td>
</tr>
<tr>
<td>13</td>
<td>Settings on the local OS/2 or Windows system for a remote queue manager platform</td>
<td>132</td>
</tr>
<tr>
<td>14</td>
<td>Default outstanding connection requests on OS/2 and Windows</td>
<td>139</td>
</tr>
<tr>
<td>15</td>
<td>Configuration worksheet for Communications Manager/2</td>
<td>144</td>
</tr>
<tr>
<td>16</td>
<td>Configuration worksheet for MQSeries for OS/2 Warp</td>
<td>161</td>
</tr>
<tr>
<td>17</td>
<td>Configuration worksheet for IBM Communications Server for Windows systems</td>
<td>168</td>
</tr>
<tr>
<td>18</td>
<td>Configuration worksheet for WebSphere MQ for Windows</td>
<td>184</td>
</tr>
<tr>
<td>19</td>
<td>Default outstanding connection requests</td>
<td>193</td>
</tr>
<tr>
<td>20</td>
<td>Settings on the local UNIX system for a remote queue manager platform</td>
<td>194</td>
</tr>
<tr>
<td>21</td>
<td>Configuration worksheet for Communications Server for AIX</td>
<td>197</td>
</tr>
<tr>
<td>22</td>
<td>Configuration worksheet for WebSphere MQ for AIX</td>
<td>210</td>
</tr>
<tr>
<td>23</td>
<td>Configuration worksheet for MQSeries for Compaq Tru64 UNIX</td>
<td>216</td>
</tr>
<tr>
<td>24</td>
<td>Configuration worksheet for HP SNAplus2</td>
<td>219</td>
</tr>
<tr>
<td>25</td>
<td>Configuration worksheet for WebSphere MQ for HP-UX</td>
<td>239</td>
</tr>
<tr>
<td>26</td>
<td>Configuration worksheet for AT&amp;T GTS SNA Services</td>
<td>243</td>
</tr>
<tr>
<td>27</td>
<td>Configuration worksheet for MQSeries for AT&amp;T GTS UNIX</td>
<td>253</td>
</tr>
<tr>
<td>28</td>
<td>Configuration worksheet for SunLink Version 9.1</td>
<td>258</td>
</tr>
<tr>
<td>29</td>
<td>Configuration worksheet for SNAP-IX</td>
<td>272</td>
</tr>
<tr>
<td>30</td>
<td>Configuration worksheet for WebSphere MQ for Solaris</td>
<td>290</td>
</tr>
<tr>
<td>31</td>
<td>Channel tasks</td>
<td>344</td>
</tr>
<tr>
<td>32</td>
<td>Settings on the local z/OS system for a remote queue manager platform</td>
<td>363</td>
</tr>
<tr>
<td>33</td>
<td>Configuration worksheet for z/OS using LU 6.2</td>
<td>366</td>
</tr>
<tr>
<td>34</td>
<td>Configuration worksheet for WebSphere MQ for z/OS</td>
<td>372</td>
</tr>
<tr>
<td>35</td>
<td>Program and transaction names</td>
<td>384</td>
</tr>
<tr>
<td>36</td>
<td>Message Channel List menu-bar choices</td>
<td>386</td>
</tr>
<tr>
<td>37</td>
<td>Menu-bar choices on channel panels</td>
<td>404</td>
</tr>
<tr>
<td>38</td>
<td>Channel attribute fields per channel type</td>
<td>406</td>
</tr>
<tr>
<td>39</td>
<td>Settings for LU 6.2 TP name on the local z/OS system for a remote queue manager platform</td>
<td>406</td>
</tr>
<tr>
<td>40</td>
<td>Configuration worksheet for z/OS using LU 6.2</td>
<td>422</td>
</tr>
<tr>
<td>41</td>
<td>Configuration worksheet for WebSphere MQ for z/OS using CICS</td>
<td>427</td>
</tr>
<tr>
<td>42</td>
<td>Configuration worksheet for z/OS using LU 6.2</td>
<td>448</td>
</tr>
<tr>
<td>43</td>
<td>Configuration worksheet for WebSphere MQ for z/OS using queue-sharing groups</td>
<td>454</td>
</tr>
<tr>
<td>44</td>
<td>Channel attribute fields per message channel type</td>
<td>498</td>
</tr>
<tr>
<td>45</td>
<td>Program and transaction names</td>
<td>507</td>
</tr>
<tr>
<td>46</td>
<td>Channel states on OS/400</td>
<td>508</td>
</tr>
<tr>
<td>47</td>
<td>Settings on the local OS/400 system for a remote queue manager platform</td>
<td>513</td>
</tr>
<tr>
<td>48</td>
<td>Configuration worksheet for SNA on an iSeries system</td>
<td>521</td>
</tr>
<tr>
<td>49</td>
<td>Configuration worksheet for WebSphere MQ for iSeries</td>
<td>535</td>
</tr>
<tr>
<td>50</td>
<td>Configuration worksheet for VSE/ESA using APPC</td>
<td>549</td>
</tr>
<tr>
<td>51</td>
<td>Configuration worksheet for MQSeries for VSE/ESA</td>
<td>554</td>
</tr>
<tr>
<td>52</td>
<td>Channel exits available for each channel type</td>
<td>571</td>
</tr>
<tr>
<td>53</td>
<td>Identifying API calls</td>
<td>580</td>
</tr>
<tr>
<td>54</td>
<td>C header files</td>
<td>612</td>
</tr>
<tr>
<td>55</td>
<td>COBOL COPY files</td>
<td>612</td>
</tr>
<tr>
<td>56</td>
<td>PL/I INCLUDE files</td>
<td>612</td>
</tr>
<tr>
<td>57</td>
<td>RPG (ILE) COPY files</td>
<td>612</td>
</tr>
<tr>
<td>58</td>
<td>RPG (OPM) COPY files</td>
<td>612</td>
</tr>
<tr>
<td>59</td>
<td>System/390® assembler macros</td>
<td>613</td>
</tr>
<tr>
<td>60</td>
<td>Fields in MQCD</td>
<td>626</td>
</tr>
<tr>
<td>61</td>
<td>Fields in MQCXF</td>
<td>666</td>
</tr>
<tr>
<td>62</td>
<td>Fields in MQTXP</td>
<td>681</td>
</tr>
<tr>
<td>63</td>
<td>Fields in MQXWD</td>
<td>685</td>
</tr>
<tr>
<td>64</td>
<td>Channel planning form</td>
<td>701</td>
</tr>
<tr>
<td>65</td>
<td>Channel planning form</td>
<td>702</td>
</tr>
</tbody>
</table>
About this book

This book describes intercommunication between WebSphere MQ products. It introduces the concepts of intercommunication; transmission queues, message channel agent programs, and communication links, that are brought together to form message channels. It describes how geographically separated queue managers are linked together by message channels to form a network of queue managers. It discusses the distributed-queuing management (DQM) facility of IBM® WebSphere MQ, which provides the services that enable applications to communicate via queue managers.

DQM provides communications that conform to the WebSphere MQ Message Channel Protocol. Each WebSphere MQ product has its own implementation of this specification, and this book is concerned with these implementations.

Who this book is for

This book is for anyone needing a description of DQM. In addition, the following readers are specifically addressed:

- Network planners responsible for designing the overall queue manager network.
- Local channel planners responsible for implementing the network plan on one node.
- Application programmers responsible for designing applications that include processes, queues, and channels, perhaps without the assistance of a systems administrator.
- Systems administrators responsible for monitoring the local system, controlling exception situations, and implementing some of the planning details.
- System programmers with responsibility for designing and programming the user exits.

What you need to know to understand this book

To use and control DQM you need to have a good knowledge of WebSphere MQ in general. You also need to understand the WebSphere MQ products for the specific platforms you will be using, and the communications protocols that will be used on those platforms.
How to use this book

This book has the following parts:

Part 1, “Introduction” on page 1
Introduces the concepts of WebSphere MQ intercommunication.

This part of the book introduces WebSphere MQ intercommunication. The description in this part is general, and is not restricted to a particular platform or system.

Note: Some references are made to individual WebSphere MQ products. Details are given only for the products that this edition of the book applies to (see the edition notice for information about which WebSphere MQ products these are).

Part 2, “How intercommunication works” on page 33
Describes the functions performed by the distributed-queuing management (DQM) facilities. Read this part to understand DQM’s role in the context of WebSphere MQ.

Part 3, “DQM in WebSphere MQ for UNIX systems, and Windows systems, and MQSeries for Compaq OpenVMS Alpha, Compaq NonStop Kernel, and OS/2 Warp” on page 105
Is specific to WebSphere MQ products on distributed platforms. It helps you to install and customize DQM on these platforms. It explains how to establish message channels to other systems and how to manage and control them.

Part 4, “DQM in WebSphere MQ for z/OS” on page 337
Is specific to WebSphere MQ for z/OS. It helps you to install and customize DQM. It explains how to establish message channels to other systems and how to manage and control them.

Part 5, “DQM in WebSphere MQ for iSeries” on page 483
Is specific to WebSphere MQ for iSeries. It helps you to install and customize DQM. It explains how to establish message channels to other systems and how to manage and control them.

Part 6, “DQM in MQSeries for VSE/ESA” on page 547
Is specific to WebSphere MQ for VSE/ESA™. It contains an example of how to set up communication to other systems.

Part 7, “Further intercommunication considerations” on page 567
Tells you about channel exit programs, which are an optional feature of DQM that allow you to add your own facilities to distributed queuing. It gives some guidance on the problems you may experience, how to recognize these problems, and what to do about them.

The Appendixes contain extra information that is pertinent to DQM:

Appendix A, “Channel planning form” gives an explanation of one suggested method of planning and maintaining DQM objects and channels.

Appendix B, “Constants for channels and exits” gives the values of named constants that apply to the channels and exits in the MQI that are discussed in this book.

Appendix C, “Queue name resolution” provides a detailed description of name resolution by queue managers. You need to understand this process in order to take full advantage of DQM.
Appendix D, “Configuration file stanzas for distributed queuing” gives information about the configuration file stanzas that relate to distributed queuing.

Appearance of text in this book

This book uses the following type styles:

CompCode
Example of the name of a parameter of a call

Terms used in this book

In the body of this book, the following shortened names are used:

CICS® The CICS Transaction Server for z/OS® (CICS/Enterprise Systems Architecture) product. (Note that, unlike other WebSphere MQ books, this book does not use the term generically to include other CICS products such as CICS for VSE/ESA.)

OS/2 OS/2 Warp

z/OS z/OS means any release of z/OS or OS/390 that supports the current version of WebSphere MQ for z/OS.

The term “UNIX systems” is used to denote the following UNIX operating systems:

- AIX
- AT&T GIS UNIX
- Compaq Tru64 UNIX
- HP-UX
- SINIX and DC/OSx
- Sun Solaris (SPARC and Intel Platform Editions)

Throughout this book, the name mqmtop has been used to represent the name of the base directory where WebSphere MQ is installed on UNIX systems.

- For AIX, the name of the actual directory is /usr/mqm
- For other UNIX systems, the name of the actual directory is /opt/mqm
Summary of changes

This section describes changes in this edition of WebSphere MQ Intercommunication. Changes since the previous edition of the book are marked by vertical lines to the left of the changes.

Changes for this edition (SC34–6059–00)

The major changes to WebSphere MQ Intercommunication include:

- Secure Sockets Layer (SSL)
- New channel attributes
- Improvements to Part 4, “DQM in WebSphere MQ for z/OS” on page 337

Changes for the previous edition (SC33–1872–05)

This edition of MQSeries Intercommunication deals with the following new features for MQSeries for OS/390 Version 5.2:

- Chapter 23, ‘Distributed queuing with queue-sharing groups’
- Chapter 24, ‘Intra-group queuing’.

The effects of these new features on monitoring, preparing and planning for MQSeries for OS/390 are dealt with in the other chapters of Part 4, ‘DQM in MQSeries for OS/390’ that cover MQSeries for OS/390 without CICS.

This edition also includes revisions to Part 1, ‘Introduction’ and Part 2, ‘MQSeries distributed-messaging techniques’.
### Part 1. Introduction

#### Chapter 1. Concepts of intercommunication

- What is intercommunication?  
- How does distributed queuing work?  
- What do we call the components?  
- Components needed to send a message  
- Components needed to return a message  
- Cluster components  
- Distributed queuing components  
  - Message channels  
  - Sender-receiver channels  
  - Requester-server channel  
  - Requester-sender channel  
  - Server-receiver channel  
  - Cluster-sender channels  
  - Cluster-receiver channels  
  - Message channel agents  
  - Transmission queues  
  - Channel initiators and listeners  
  - Channel-exit programs  
- Dead-letter queues  
- Remote queue definitions  
- How to get to the remote queue manager  
- Multi-hopping  
- Sharing channels  
- Using different channels  
- Using clustering

#### Chapter 2. Making your applications communicate

- How to send a message to another queue manager  
  - Defining the channels  
  - Defining the queues  
  - Sending the messages  
  - Starting the channel  
  - Triggering channels  
  - Safety of messages  
    - Fast, nonpersistent messages  
    - Undelivered messages

#### Chapter 3. More about intercommunication

- Addressing information  
- What are aliases?  
- Queue name resolution  
- Queue manager alias definitions  
  - Outbound messages - remapping the queue manager name  
  - Outbound messages - altering or specifying the transmission queue  
  - Inbound messages - determining the destination  
- Reply-to queue alias definitions  
  - What is a reply-to queue alias definition?  
  - Reply-to queue name  
- Networks  
  - Channel and transmission queue names  
  - Network planner
Introduction
Chapter 1. Concepts of intercommunication

This chapter introduces the concepts of intercommunication in WebSphere MQ.

- The basic concepts of intercommunication are explained in “What is intercommunication?”
- The objects that you need for intercommunication are described in “Distributed queuing components” on page 7

This chapter goes on to introduce:

- “Dead-letter queues” on page 13
- “Remote queue definitions” on page 14
- “How to get to the remote queue manager” on page 14

What is intercommunication?

In WebSphere MQ, intercommunication means sending messages from one queue manager to another. The receiving queue manager could be on the same machine or another; nearby or on the other side of the world. It could be running on the same platform as the local queue manager, or could be on any of the platforms supported by WebSphere MQ. This is called a distributed environment. WebSphere MQ handles communication in a distributed environment such as this using Distributed Queue Management (DQM).

The local queue manager is sometimes called the source queue manager and the remote queue manager is sometimes called the target queue manager or the partner queue manager.

How does distributed queuing work?

Figure 1 on page 4 shows an overview of the components of distributed queuing.
1. An application uses the MQCONN call to connect to a queue manager.
2. The application then uses the MQOPEN call to open a queue so that it can put messages on it.
3. A queue manager has a definition for each of its queues, specifying information such as the maximum number of messages allowed on the queue.
4. If the messages are destined for a queue on a remote system, the local queue manager holds them in a message store until it is ready to forward them to the remote queue manager. This can be transparent to the application.
5. Each queue manager contains communications software called the *moving service* component; through this, the queue manager can communicate with other queue managers.
6. The *transport service* is independent of the queue manager and can be any one of the following (depending on the platform):
   - Systems Network Architecture Advanced Program-to Program Communication (SNA APPC)
   - Transmission Control Protocol/Internet Protocol (TCP/IP)
   - Network Basic Input/Output System (NetBIOS)
   - Sequenced Packet Exchange (SPX)
   - User-Datagram Protocol (UDP)

What do we call the components?

1. WebSphere MQ applications can put messages onto a local queue, that is, a queue on the queue manager the application is connected to.
2. A queue manager has a definition for each of its queues. It can also have definitions for queues that are owned by other queue managers. These are called *remote queue definitions*. WebSphere MQ applications can also put messages targeted at these remote queues.
3. If the messages are destined for a remote queue manager, the local queue manager stores them on a *transmission queue* until it is ready to send them to the remote queue manager. A transmission queue is a special type of local queue.
queue on which messages are stored until they can be successfully transmitted and stored at the remote queue manager.

4. The software that handles the sending and receiving of messages is called the Message Channel Agent (MCA).

5. Messages are transmitted between queue managers on a channel. A channel is a one-way communication link between two queue managers. It can carry messages destined for any number of queues at the remote queue manager.

**Components needed to send a message**

If a message is to be sent to a remote queue manager, the local queue manager needs definitions for a transmission queue and a channel.

Each end of a channel has a separate definition, defining it, for example, as the sending end or the receiving end. A simple channel consists of a *sender* channel definition at the local queue manager and a *receiver* channel definition at the remote queue manager. These two definitions must have the same name, and together constitute one channel.

There is also a *message channel agent* (MCA) at each end of a channel.

Each queue manager should have a *dead-letter queue* (also known as the *undelivered message queue*). Messages are put on this queue if they cannot be delivered to their destination.

**Components needed to return a message**

If your application requires messages to be returned from the remote queue manager, you need to define another channel, to run in the opposite direction between the queue managers, as shown in **Figure 3 on page 6**.

**Figure 2** shows the relationship between queue managers, transmission queues, channels, and MCAs.
Cluster components
An alternative to the traditional WebSphere MQ network is the use of clusters. Clusters are supported on WebSphere MQ for AIX, iSeries, HP-UX, Solaris, z/OS, and Windows, and MQSeries V5.1 for Compaq Tru64 UNIX, and OS/2 Warp only.

A cluster is a network of queue managers that are logically associated in some way. You can group queue managers in a cluster so that queue managers can make the queues that they host available to every other queue manager in the cluster. Assuming you have the necessary network infrastructure in place, any queue manager can send a message to any other queue manager in the same cluster without the need for explicit channel definitions, remote-queue definitions, or transmission queues for each destination. Every queue manager in a cluster has a single transmission queue that transmits messages to any other queue manager in the cluster. Each queue manager needs to define only one cluster-receiver channel and one cluster-sender channel.

Figure 4 on page 7 shows the components of a cluster called CLUSTER:
CLUSTER contains three queue managers, QM1, QM2, and QM3.
QM1 and QM2 host full repositories of information about the queue managers and queues in the cluster.
QM2 and QM3 host some cluster queues, that is, queues that are accessible to any other queue manager in the cluster.
Each queue manager has a cluster-receiver channel called TO.qmgr on which it can receive messages.
Each queue manager also has a cluster-sender channel on which it can send information to one of the repository queue managers.
QM1 and QM3 send to the repository at QM2 and QM2 sends to the repository at QM1.

As with distributed queuing, you use the MQPUT call to put a message to a queue at any queue manager. You use the MQGET call to retrieve messages from a local queue.

For further information about clusters, see the WebSphere MQ Queue Manager Clusters book.

Distributed queuing components

This section describes the components of distributed queuing. These are:
- Message channels
- Message channel agents
- Transmission queues
- Channel initiators and listeners
- Channel-exit programs
Message channels

Message channels are the channels that carry messages from one queue manager to another.

Do not confuse message channels with MQI channels. There are two types of MQI channel, server-connection and client-connection. These are discussed in the WebSphere MQ Clients book.

The definition of each end of a message channel can be one of the following types:
- Sender
- Receiver
- Server
- Requester
- Cluster sender
- Cluster receiver

A message channel is defined using one of these types defined at one end, and a compatible type at the other end. Possible combinations are:
- Sender-receiver
- Requester-server
- Requester-sender (callback)
- Server-receiver
- Cluster sender-cluster receiver

Sender-receiver channels
A sender in one system starts the channel so that it can send messages to the other system. The sender requests the receiver at the other end of the channel to start. The sender sends messages from its transmission queue to the receiver. The receiver puts the messages on the destination queue. Figure 5 illustrates this.

Requester-server channel
A requester in one system starts the channel so that it can receive messages from the other system. The requester requests the server at the other end of the channel to start. The server sends messages to the requester from the transmission queue defined in its channel definition.

A server channel can also initiate the communication and send messages to a requester, but this applies only to fully qualified servers, that is server channels that have the connection name of the partner specified in the channel definition. A fully qualified server can either be started by a requester, or can initiate a communication with a requester.
Requester-sender channel
The requester starts the channel and the sender terminates the call. The sender then restarts the communication according to information in its channel definition (this is known as callback). It sends messages from the transmission queue to the requester.

Server-receiver channel
This is similar to sender-receiver but applies only to fully qualified servers, that is server channels that have the connection name of the partner specified in the channel definition. Channel startup must be initiated at the server end of the link. The illustration of this is similar to the illustration in Figure 5 on page 8.

Cluster-sender channels
In a cluster, each queue manager has a cluster-sender channel on which it can send cluster information to one of the full repository queue managers. Queue managers can also send messages to other queue managers on cluster-sender channels.
Cluster-receiver channels
In a cluster, each queue manager has a cluster-receiver channel on which it can receive messages and information about the cluster. The illustration of this is similar to the illustration in Figure 8.

Message channel agents
A message channel agent (MCA) is a program that controls the sending and receiving of messages. There is one message channel agent at each end of a channel. One MCA takes messages from the transmission queue and puts them on the communication link. The other MCA receives messages and delivers them onto a queue on the remote queue manager.

A message channel agent is called a caller MCA if it initiated the communication, otherwise it is called a responder MCA. A caller MCA may be associated with a sender, cluster-sender, server (fully qualified), or requester channel. A responder MCA may be associated with any type of message channel, except a cluster sender.

Transmission queues
A transmission queue is a special type of local queue used to store messages before they are transmitted by the MCA to the remote queue manager. In a distributed-queuing environment, you need to define one transmission queue for each sending MCA, unless you are using WebSphere MQ Queue Manager clusters.

You specify the name of the transmission queue in a remote queue definition, (see “Remote queue definitions” on page 14). If you do not specify the name, the queue manager looks for a transmission queue with the same name as the remote queue manager.

You can specify the name of a default transmission queue for the queue manager. This is used if you do not specify the name of the transmission queue, and a transmission queue with the same name as the remote queue manager does not exist.

Channel initiators and listeners
A channel initiator acts as a trigger monitor for sender channels, because a transmission queue may be defined as a triggered queue. When a message arrives on a transmission queue that satisfies the triggering criteria for that queue, a message is sent to the initiation queue, triggering the channel initiator to start the
appropriate sender channel. You can also start server channels in this way if you specified the connection name of the partner in the channel definition. This means that channels can be started automatically, based upon messages arriving on the appropriate transmission queue.

You need a channel listener program to start receiving (responder) MCAs. Responder MCAs are started in response to a startup request from the caller MCA; the channel listener detects incoming network requests and starts the associated channel.

Figure 9 shows how channel initiators and channel listeners are used.

The implementation of channel initiators is platform specific.
• In z/OS native distributed queuing, there is one channel initiator for each queue manager and it runs as a separate address space. You start it using the WebSphere MQ command START CHINIT, which you would normally issue as part of your queue manager startup. It monitors the system-defined queue SYSTEM.CHANNEL.INITQ, which is the initiation queue that is recommended for all the transmission queues.
• On z/OS, if you are using CICS for distributed queuing, there is no channel initiator. To implement triggering, use the CICS trigger monitor transaction, CKTI, to monitor the initiation queue.
• On other platforms, you can start as many channel initiators as you like, specifying a name for the initiation queue for each one. Normally you need only one initiator. WebSphere MQ for AIX, iSeries, HP-UX, Solaris and Windows systems, and MQSeries for Compaq Tru64 UNIX, and OS/2 Warp allow you to start up to three channel initiators (the default value), but you can change this value. On platforms that support clustering, when you start a queue manager, a channel initiator also is automatically started.

The channel initiator is also required for other functions. These are discussed later in this book.

The implementation of channel listeners is platform specific.
• Use the channel listener programs provided by WebSphere MQ if you are using z/OS native distributed queuing, MQSeries for Compaq (DIGITAL) Open VMS, and Compaq NonStop Kernel.
Distributed queuing components

Note: On z/OS, The TCP/IP listener can be started many times with different combinations of port number and address to listen on. For more information, see "Listeners" on page 439.

- If you are using CICS for distributed queuing on z/OS, you do not need a channel listener because CICS provides this function.
- On iSeries, use the channel listener program provided by WebSphere MQ if you are using TCP/IP. If you are using SNA, you do not need a listener program. SNA starts the channel by invoking the receiver program on the remote system.
- On OS/2 and Windows systems, you can use either the channel listener program provided by WebSphere MQ, or the facilities provided by the 'operating system' (for example, Attach manager for LU 6.2 communications on OS/2). If performance is important in your environment and if the environment is stable, you can choose to run the WebSphere MQ listener as a trusted application as described in "Running channels and listeners as trusted applications" on page 126. See the WebSphere MQ Application Programming Guide for information about trusted applications.
- On UNIX systems, use the channel listener program provided by WebSphere MQ or the facilities provided by the 'operating system' (for example, inetd for TCP/IP communications).

Channel-exit programs

If you want to do some additional processing (for example, encryption or data compression) you can write your own channel-exit programs, or sometimes use SupportPacs. The Transaction Processing SupportPacs library for WebSphere MQ is available on the Internet at URL:


WebSphere MQ calls channel-exit programs at defined places in the processing carried out by the MCA. There are six types of channel exit:

Security exit
Used for security checking, such as authentication of the partner.

Message exit
Used for operations on the message, for example, encryption prior to transmission.

Send and receive exits
Used for operations on split messages, for example, data compression and decompression.

Message-retry exit
Used when there is a problem putting the message to the destination.

Channel auto-definition exit
Used to modify the supplied default definition for an automatically defined receiver or server-connection channel.

Transport-retry exit
Used to suspend data being sent on a channel when communication is not possible.

The sequence of processing is as follows:
1. The security exits are called after the initial data negotiation between both ends of the channel. These must end successfully for the startup phase to complete and to allow messages to be transferred.
2. The message exit is called by the sending MCA, and then the send exit is called for each part of the message that is transmitted to the receiving MCA.

3. The receiving MCA calls the receive exit when it receives each part of the message, and then calls the message exit when the whole message has been received.

This is illustrated in Figure 10.

![Figure 10. Sequence in which channel exit programs are called](image)

The message-retry exit is used to determine how many times the receiving MCA will attempt to put a message to the destination queue before taking alternative action. It is not supported on WebSphere MQ for z/OS.

For more information about channel exits, see Chapter 45, “Channel-exit programs” on page 571.

**Dead-letter queues**

The dead-letter queue (or undelivered-message queue) is the queue to which messages are sent if they cannot be routed to their correct destination. Messages are put on this queue when they cannot be put on the destination queue for some reason (for example, because the queue does not exist, or because it is full). Dead-letter queues are also used at the sending end of a channel, for data-conversion errors.

We recommend that you define a dead-letter queue for each queue manager. If you do not, and the MCA is unable to put a message, it is left on the transmission queue and the channel is stopped.

Also, if fast, non-persistent messages (see Fast, nonpersistent messages on page 22) cannot be delivered and no DLQ exists on the target system, these messages are discarded.

However, using dead-letter queues can affect the sequence in which messages are delivered, and so you may choose not to use them.
Remote queue definitions

Whereas applications can retrieve messages only from local queues, they can put messages on local queues or remote queues. Therefore, as well as a definition for each of its local queues, a queue manager may have remote queue definitions. These are definitions for queues that are owned by another queue manager. The advantage of remote queue definitions is that they enable an application to put a message to a remote queue without having to specify the name of the remote queue or the remote queue manager, or the name of the transmission queue. This gives you location independence.

There are other uses for remote queue definitions, which will be described later.

How to get to the remote queue manager

You may not always have one channel between each source and target queue manager. Consider these alternative possibilities.

Multi-hopping

If there is no direct communication link between the source queue manager and the target queue manager, it is possible to pass through one or more intermediate queue managers on the way to the target queue manager. This is known as a multi-hop.

You need to define channels between all the queue managers, and transmission queues on the intermediate queue managers. This is shown in Figure 11.

Figure 11. Passing through intermediate queue managers

Sharing channels

As an application designer, you have the choice of forcing your applications to specify the remote queue manager name along with the queue name, or creating a remote queue definition for each remote queue. This definition holds the remote queue manager name, the queue name, and the name of the transmission queue. Either way, all messages from all applications addressing queues at the same
remote location have their messages sent through the same transmission queue. This is shown in Figure 12.

Figure 12. Sharing a transmission queue

Figure 12 illustrates that messages from multiple applications to multiple remote queues can use the same channel.

Using different channels

If you have messages of different types to send between two queue managers, you can define more than one channel between the two. There are times when you need alternative channels, perhaps for security purposes, or to trade off delivery speed against sheer bulk of message traffic.

To set up a second channel you need to define another channel and another transmission queue, and create a remote queue definition specifying the location and the transmission queue name. Your applications can then use either channel but the messages will still be delivered to the same target queues. This is shown in Figure 13.

Figure 13. Using multiple channels

When you use remote queue definitions to specify a transmission queue, your applications must not specify the location (that is, the destination queue manager) themselves. If they do, the queue manager will not make use of the remote queue definitions. Remote queue definitions make the location of queues and the transmission queue transparent to applications. Applications can put messages to a
Getting to remote queue manager

A logical queue without knowing where the queue is located and you can alter the physical queue without having to change your applications.

Using clustering

Every queue manager within a cluster defines a cluster-receiver channel. When another queue manager wants to send a message to that queue manager, it defines the corresponding cluster-sender channel automatically. For example, if there is more than one instance of a queue in a cluster, the cluster-sender channel could be defined to any of the queue managers that host the queue. WebSphere MQ uses a workload management algorithm that uses a round-robin routine to select an available queue manager to route a message to. For more information see the *WebSphere MQ Queue Manager Clusters* book.
Chapter 2. Making your applications communicate

This chapter provides more detailed information about intercommunication between WebSphere MQ products. Before reading this chapter it is helpful to have an understanding of channels, queues, and the other concepts introduced in Chapter 1, “Concepts of intercommunication” on page 3.

This chapter covers the following topics:

- “How to send a message to another queue manager”
- “Triggering channels” on page 20
- “Safety of messages” on page 22

How to send a message to another queue manager

This section describes the simplest way to send a message from one queue manager to another.

Before you do this you need to do the following:
1. Check that your chosen communication protocol is available.
2. Start the queue managers.
3. Start the channel initiators.
4. Start the listeners.

You also need to have the correct WebSphere MQ security authorization to create the objects required.

To send messages from one queue manager to another:

- Define the following objects on the source queue manager:
  - Sender channel
  - Remote queue definition
  - Initiation queue (required on z/OS, otherwise optional)
  - Transmission queue
  - Dead-letter queue (recommended)
- Define the following objects on the target queue manager:
  - Receiver channel
  - Target queue
  - Dead-letter queue (recommended)

You can use several different methods to define these objects, depending on your WebSphere MQ platform:

z/OS or MVS/ESA™

If you are using z/OS native distributed queuing, you can use the Operation and Control panels (see the WebSphere MQ System Administration Guide for more information) WebSphere MQ script commands (MQSC). See the WebSphere MQ Script (MQSC) Command Reference for more information.

If you are using z/OS distributed queuing with CICS, you must use the supplied CICS transaction CKMC for channels.

iSeries

You can use the panel interface, the WebSphere MQ script commands described in the WebSphere MQ Script (MQSC) Command Reference book, or
Sending messages

the programmable command format (PCF) commands described in the
WebSphere MQ Programmable Command Formats and Administration Interface
book.

OS/2, Windows systems, UNIX systems, and Compaq OpenVMS Alpha
You can use the WebSphere MQ commands described in the WebSphere MQ
Script (MQSC) Command Reference book, or the PCF commands described in
the WebSphere MQ Programmable Command Formats and Administration
Interface book. On Windows systems only, you can also use the graphical
user interfaces, the WebSphere MQ explorer and the WebSphere MQ Web
Administration.

Compaq NonStop Kernel
You can use MQSC commands, PCF commands, or the Message Queue
Management facility. See the MQSeries for Compaq NonStop Kernel System
Administration for more information about the control commands and the
Message Queue Management facility.

VSE/ESA
You can use the panel interface as described in the MQSeries for VSE/ESA
System Management Guide.

The different methods are described in more detail in the platform-specific parts of
this book.

Defining the channels

To send messages from one queue manager to another, you need to define two
channels; one on the source queue manager and one on the target queue manager.

On the source queue manager
Define a channel with a channel type of SENDER. You need to specify the
following:

• The name of the transmission queue to be used (the XMITQ attribute).
• The connection name of the partner system (the CONNAME attribute).
• The name of the communication protocol you are using (the TRPTYPE
attribute). On AIX, iSeries, Compaq Tru64 UNIX, HP-UX, OS/2 Warp,
Solaris, and Windows, you do not have to specify this. You can leave it
to pick up the value from your default channel definition. On MQSeries
for VSE/ESA, the protocol must be TCP or LU 6.2; you can choose T or
L accordingly on the Maintain Channel Definition menu. On
WebSphere MQ for z/OS, the protocol must be TCP or LU6.2.

Details of all the channel attributes are given in Chapter 6, “Channel
attributes” on page 77.

On the target queue manager
Define a channel with a channel type of RECEIVER, and the same name as
the sender channel.

Specify the name of the communication protocol you are using (the
TRPTYPE attribute). For WebSphere MQ for AIX, iSeries, HP-UX, Solaris
and Windows systems, and MQSeries for Compaq Tru64 UNIX, and OS/2
Warp, you do not have to specify this. You can leave it to pick up the
value from your default channel definition. If you are using CICS to define
a channel, you cannot specify TRPTYPE. Instead you should accept the
defaults provided. On MQSeries for VSE/ESA, you can choose T (TCP) or
L (LU 6.2) on the Maintain Channel Definition menu. On WebSphere MQ
for z/OS, the protocol must be TCP or LU6.2.
Note that receiver channel definitions can be generic. This means that if you have several queue managers communicating with the same receiver, the sending channels can all specify the same name for the receiver, and one receiver definition will apply to them all.

When you have defined the channel, you can test it using the PING CHANNEL command. This command sends a special message from the sender channel to the receiver channel and checks that it is returned.

## Defining the queues

To send messages from one queue manager to another, you need to define up to six queues; four on the source queue manager and two on the target queue manager.

### On the source queue manager

- Remote queue definition
  
  In this definition you specify the following:
  
  **Remote queue manager name**
  
  The name of the target queue manager.

  **Remote queue name**
  
  The name of the target queue on the target queue manager.

  **Transmission queue name**
  
  The name of the transmission queue. You do not have to specify this. If you do not, a transmission queue with the same name as the target queue manager is used, or, if this does not exist, the default transmission queue is used. You are advised to give the transmission queue the same name as the target queue manager so that the queue is found by default.

- Initiation queue definition
  
  Required on z/OS, and optional on other platforms. On z/OS you must use the initiation queue called SYSTEM.CHANNEL.INITQ and you are recommended to do so on other platforms also.

- Transmission queue definition
  
  A local queue with the USAGE attribute set to XMITQ. If you are using the WebSphere MQ for iSeries native interface, the USAGE attribute is "TMQ.

- Dead-letter queue definition—recommended
  
  Define a dead-letter queue to which undelivered messages can be written.

### On the target queue manager

- Local queue definition
  
  The target queue. The name of this queue must be the same as that specified in the remote queue name field of the remote queue definition on the source queue manager.

- Dead-letter queue definition—recommended
  
  Define a dead-letter queue to which undelivered messages can be written.
Sending messages

Sending the messages

When you put messages on the remote queue defined at the source queue manager, they are stored on the transmission queue until the channel is started. When the channel has been started, the messages are delivered to the target queue on the remote queue manager.

Starting the channel

Start the channel on the sending queue manager using the START CHANNEL command. When you start the sending channel, the receiving channel is started automatically (by the listener) and the messages are sent to the target queue. Both ends of the message channel must be running for messages to be transferred.

Because the two ends of the channel are on different queue managers, they could have been defined with different attributes. To resolve any differences, there is an initial data negotiation between the two ends when the channel starts. In general, the two ends of the channel agree to operate with the attributes needing the fewer resources, thus enabling larger systems to accommodate the lesser resources of smaller systems at the other end of the message channel.

The sending MCA splits large messages before sending them across the channel. They are reassembled at the remote queue manager. This is transparent to the user.

An MCA can transfer messages using multiple threads. This process, called pipelining enables the MCA to transfer messages more efficiently, with fewer wait states. This improves channel performance.

Triggering channels

This explanation is intended as an overview of triggering concepts. You can find a complete description in the WebSphere MQ Application Programming Guide.

For platform-specific information see the following:
• For OS/2, Windows systems, UNIX systems, Compaq OpenVMS Alpha, and Compaq NonStop Kernel, “Triggering channels” on page 121
• For z/OS native distributed queuing, “Defining WebSphere MQ objects” on page 357
• For z/OS distributed queuing with CICS, “How to trigger channels” on page 390
• For iSeries, “Triggering channels” on page 505
The objects required for triggering are shown in Figure 14. It shows the following sequence of events:

1. The local queue manager places a message from an application or from a message channel agent (MCA) on the transmission queue.
2. When the triggering conditions are fulfilled, the local queue manager places a trigger message on the initiation queue.
3. The long-running channel initiator program monitors the initiation queue, and retrieves messages as they appear.
4. The channel initiator processes the trigger messages according to information contained in them. This information may include the channel name, in which case the corresponding MCA is started.
5. The local application or the MCA, having been triggered, retrieves the messages from the transmission queue.

To set up this scenario, you need to:

- Create the transmission queue with the name of the initiation queue (that is, SYSTEM.CHANNEL.INITQ) in the corresponding attribute.
- Ensure that the initiation queue (SYSTEM.CHANNEL.INITQ) exists.
- Ensure that the channel initiator program is available and running. The channel initiator program must be provided with the name of the initiation queue in its start command. On z/OS native distributed queuing, the name of the initiation queue is fixed, so is not used on the start command.
- Create the process definition for the triggering, if it does not exist, and ensure that its UserData field contains the name of the channel it serves. For WebSphere MQ for AIX, iSeries, HP-UX, Linux, Solaris and Windows systems, and MQSeries for Compaq Tru64 UNIX, and OS/2 Warp, the process definition is optional (it is not supported on MQSeries for VSE/ESA). Instead, you can specify the channel name in the TriggerData attribute of the transmission queue. WebSphere MQ for AIX, iSeries, Linux, HP-UX, Solaris and Windows systems,
Triggering channels

and MQSeries for Compaq Tru64 UNIX, and OS/2 Warp allow the channel name to be specified as blank, in which case the first available channel definition with this transmission queue is used.

- Ensure that the transmission queue definition contains the name of the process definition to serve it, (if applicable), the initiation queue name, and the triggering characteristics you feel are most suitable. The trigger control attribute allows triggering to be enabled, or not, as necessary.

Notes:
1. The channel initiator program acts as a ‘trigger monitor’ monitoring the initiation queue used to start channels.
2. An initiation queue and trigger process can be used to trigger any number of channels.
3. Any number of initiation queues and trigger processes can be defined.
4. A trigger type of FIRST is recommended, to avoid flooding the system with channel starts.

Safety of messages

In addition to the usual recovery features of WebSphere MQ, distributed queue management ensures that messages are delivered properly by using a syncpoint procedure coordinated between the two ends of the message channel. If this procedure detects an error, it closes the channel to allow you to investigate the problem, and keeps the messages safely in the transmission queue until the channel is restarted.

The syncpoint procedure has an added benefit in that it attempts to recover an in-doubt situation when the channel starts up. (In-doubt is the status of a unit of recovery for which a syncpoint has been requested but the outcome of the request is not yet known.) Also associated with this facility are the two functions:
1. Resolve with commit or backout
2. Reset the sequence number

The use of these functions occurs only in exceptional circumstances because the channel recovers automatically in most cases.

Fast, nonpersistent messages

In WebSphere MQ for AIX, iSeries, HP-UX, Linux, Solaris, Windows systems, and z/OS without CICS and MQSeries V5.1 for Compaq Tru64 UNIX, and OS/2 Warp, the nonpersistent message speed (NPMSPEED) channel attribute can be used to specify that any nonpersistent messages on the channel are to be delivered more quickly. For more information about this attribute, see “Nonpersistent message speed (NPMSPEED)” on page 93.

If a channel terminates while fast, nonpersistent messages are in transit, the messages may be lost and it is up to the application to arrange for their recovery if required.

If the receiving channel cannot put the message to its destination queue then it is placed on the dead letter queue, if one has been defined. If not, the message is discarded.

In MQSeries for Compaq OpenVMS Alpha fast messages are enabled differently. For channels of type sender, server, receiver or requester, set the description field at both ends of the channel as follows:
Specifying \texttt{>>>} as the first characters in the channel description defines the channel as fast for nonpersistent messages.

\textbf{Note:} If the other end of the channel does not support the option, the channel runs at normal speed.

\section*{Undelivered messages}

For information about what happens when a message cannot be delivered, see "What happens when a message cannot be delivered?" on page 71.
Chapter 3. More about intercommunication

This chapter mentions three aliases:
- Remote queue definition
- Queue manager alias definition
- Reply-to queue alias definition

These are all based on the remote queue definition object introduced in “Remote queue definitions” on page 14.

This discussion does not apply to alias queues. These are described in the WebSphere MQ Application Programming Guide.

This chapter also discusses “Networks” on page 29.

Addressing information

In a single-queue-manager environment, the address of a destination queue is established when an application opens a queue for putting messages to. Because the destination queue is on the same queue manager, there is no need for any addressing information.

In a distributed environment the queue manager needs to know not only the destination queue name, but also the location of that queue (that is, the queue manager name), and the route to that remote location (that is, the transmission queue). When an application puts messages that are destined for a remote queue manager, the local queue manager adds a transmission header to them before placing them on the transmission queue. The transmission header contains the name of the destination queue and queue manager, that is, the addressing information. The receiving channel removes the transmission header and uses the information in it to locate the destination queue.

You can avoid the need for your applications to specify the name of the destination queue manager if you use a remote queue definition. This definition specifies the name of the remote queue, the name of the remote queue manager to which messages are destined, and the name of the transmission queue used to transport the messages.

What are aliases?

Aliases are used to provide a quality of service for messages. The queue manager alias enables a system administrator to alter the name of a target queue manager without causing you to have to change your applications. It also enables the system administrator to alter the route to a destination queue manager, or to set up a route that involves passing through a number of other queue managers (multi-hopping). The reply-to queue alias provides a quality of service for replies.

Queue manager aliases and reply-to queue aliases are created using a remote-queue definition that has a blank RNAME. These definitions do not define real queues; they are used by the queue manager to resolve physical queue names, queue manager names, and transmission queues.

Alias definitions are characterized by having a blank RNAME.
Queue name resolution

Queue name resolution occurs at every queue manager each time a queue is opened. Its purpose is to identify the target queue, the target queue manager (which may be local), and the route to that queue manager (which may be null). The resolved name has three parts: the queue manager name, the queue name, and, if the queue manager is remote, the transmission queue.

When a remote queue definition exists, no alias definitions are referenced. The queue name supplied by the application is resolved to the name of the destination queue, the remote queue manager, and the transmission queue specified in the remote queue definition. For more detailed information about queue name resolution, see Appendix C, “Queue name resolution” on page 711.

If there is no remote queue definition and a queue manager name is specified, or resolved by the name service, the queue manager looks to see if there is a queue manager alias definition that matches the supplied queue manager name. If there is, the information in it is used to resolve the queue manager name to the name of the destination queue manager. The queue manager alias definition can also be used to determine the transmission queue to the destination queue manager.

If the resolved queue name is not a local queue, both the queue manager name and the queue name are included in the transmission header of each message put by the application to the transmission queue.

The transmission queue used usually has the same name as the resolved queue manager, unless changed by a remote queue definition or a queue manager alias definition. If you have not defined such a transmission queue but you have defined a default transmission queue, then this is used.

Note: Names of queue managers running on z/OS are limited to four characters.

Queue manager alias definitions

Queue manager alias definitions apply when an application that opens a queue to put a message, specifies the queue name and the queue manager name.

Queue manager alias definitions have three uses:
• When sending messages, remapping the queue manager name
• When sending messages, altering or specifying the transmission queue
• When receiving messages, determining whether the local queue manager is the intended destination for those messages

Outbound messages - remapping the queue manager name

Queue manager alias definitions can be used to remap the queue manager name specified in an MQOPEN call. For example, an MQOPEN call specifies a queue name of THISQ and a queue manager name of YOURQM. At the local queue manager there is a queue manager alias definition like this:
DEFINE QREMOTE (YOURQM) RMQNAME (REALQM)

This shows that the real queue manager to be used, when an application puts messages to queue manager YOURQM, is REALQM. If the local queue manager is REALQM, it puts the messages to the queue THISQ, which is a local queue. If the local queue manager is not called REALQM, it routes the message to a transmission queue called REALQM. The queue manager changes the transmission header to say REALQM instead of YOURQM.
Outbound messages - altering or specifying the transmission queue

Figure 15 shows a scenario where messages arrive at queue manager ‘QM1’ with transmission headers showing queue names at queue manager ‘QM3’. In this scenario, ‘QM3’ is reachable by multi-hopping through ‘QM2’.

All messages for ‘QM3’ are captured at ‘QM1’ with a queue manager alias. The queue manager alias is named ‘QM3’ and contains the definition ‘QM3 via transmission queue QM2’. The definition looks like this:

DEFINE QREMOTE (QM3) RNAME() RQMNAME(QM3) XMITQ(QM2)

The queue manager puts the messages on transmission queue ‘QM2’ but does not make any alteration to the transmission queue header because the name of the destination queue manager, ‘QM3’, does not alter.

All messages arriving at ‘QM1’ and showing a transmission header containing a queue name at ‘QM2’ are also put on the ‘QM2’ transmission queue. In this way, messages with different destinations are collected onto a common transmission queue to an appropriate adjacent system, for onward transmission to their destinations.

Inbound messages - determining the destination

A receiving MCA opens the queue referenced in the transmission header. If a queue manager alias definition exists with the same name as the queue manager referenced, then the queue manager name received in the transmission header is replaced with the RQMNAME from that definition.

This has two uses:
- Directing messages to another queue manager
- Altering the queue manager name to be the same as the local queue manager
Reply-to queue alias definitions

When an application needs to reply to a message it may look at the data in the message descriptor of the message it received to find out the name of the queue to which it should reply. It is up to the sending application to suggest where replies should be sent and to attach this information to its messages. This has to be coordinated as part of your application design.

What is a reply-to queue alias definition?

A reply-to queue alias definition specifies alternative names for the reply information in the message descriptor. The advantage of this is that you can alter the name of a queue or queue manager without having to alter your applications. Queue name resolution takes place at the sending end, before the message is put to a queue.

Note: This is an unusual use of queue-name resolution. It is the only situation in which name resolution takes place at a time when a queue is not being opened.

Normally an application specifies a reply-to queue and leaves the reply-to queue manager name blank. The queue manager fills in its own name at put time. This works well except when you want an alternate channel to be used for replies, for example, a channel that uses transmission queue ‘QM1_relief’ instead of the default return channel which uses transmission queue ‘QM1’. In this situation, the queue manager names specified in transmission-queue headers do not match “real” queue manager names but are re-specified using queue manager alias definitions. In order to return replies along alternate routes, it is necessary to map reply-to queue data as well, using reply-to queue alias definitions.

In the example in Figure 16:

1. The application puts a message using the MQPUT call and specifying the following in the message descriptor:

   ReplyToQ='Reply_to'
   ReplyToQMgr=''
Reply-to queue alias definitions

Note that ReplyToQMgr must be blank in order for the reply-to queue alias to be used.

2. You create a reply-to queue alias definition called ‘Reply_to’, which contains the name ‘Answer’, and the queue manager name ‘QM1_relief’.

   DEFINE QREMOTE ('Reply_to') RNAME ('Answer')
   RQMNAME ('QM1_relief')

3. The messages are sent with a message descriptor showing ReplyToQ=’Answer’ and ReplyToQMgr=’QM1_relief’.

4. The application specification must include the information that replies are to be found in queue ‘Answer’ rather than ‘Reply_to’.

To prepare for the replies you have to create the parallel return channel. This involves defining:

- At QM2, the transmission queue named ‘QM1_relief’
  
  ```
  DEFINE QLOCAL ('QM1_relief') USAGE(XMITQ)
  ```

- At QM1, the queue manager alias QM1_relief’
  
  ```
  DEFINE QREMOTE ('QM1_relief') RNAME() RQMNAME(QM1)
  ```

This queue manager alias terminates the chain of parallel return channels and captures the messages for QM1.

If you think you might want to do this at sometime in the future, arrange for your applications to use the alias name from the start. For now this is a normal queue alias to the reply-to queue, but later it can be changed to a queue manager alias.

Reply-to queue name

Care is needed with naming reply-to queues. The reason that an application puts a reply-to queue name in the message is that it can specify the queue to which its replies will be sent. But when you create a reply-to queue alias definition with this name, you cannot have the actual reply-to queue (that is, a local queue definition) with the same name. Therefore, the reply-to queue alias definition must contain a new queue name as well as the queue manager name, and the application specification must include the information that its replies will be found in this other queue.

The applications now have to retrieve the messages from a different queue from the one they named as the reply-to queue when they put the original message.

Networks

So far this book has covered creating channels between your system and any other system with which you need to have communications, and creating multi-hop channels to systems where you have no direct connections. The message channel connections described in the scenarios are shown as a network diagram in Figure 17 on page 30.

Channel and transmission queue names

You can give transmission queues any name you like, but to avoid confusion, you can give them the same names as the destination queue manager names, or queue manager alias names, as appropriate, to associate them with the route they use.

This gives a clear overview of parallel routes that you create through intermediate (multi-hopped) queue managers.
This is not quite so clear-cut for channel names. The channel names in Figure 17 for QM2, for example, must be different for incoming and outgoing channels. All channel names may still contain their transmission queue names, but they must be qualified to make them unique.

For example, at QM2, there is a QM3 channel coming from QM1, and a QM3 channel going to QM3. To make the names unique, the first one may be named ‘QM3_from_QM1’, and the second may be named ‘QM3_from_QM2’. In this way, the channel names show the transmission queue name in the first part of the name, and the direction and adjacent queue manager name in the second part of the name.

A table of suggested channel names for Figure 17 is given in Table 1.

Table 1. Example of channel names

<table>
<thead>
<tr>
<th>Route name</th>
<th>Queue managers hosting channel</th>
<th>Transmission queue name</th>
<th>Suggested channel name</th>
</tr>
</thead>
<tbody>
<tr>
<td>QM1</td>
<td>QM1 &amp; QM2</td>
<td>QM1 (at QM2)</td>
<td>QM1_from_QM2</td>
</tr>
<tr>
<td>QM1</td>
<td>QM2 &amp; QM3</td>
<td>QM1 (at QM3)</td>
<td>QM1_from_QM3</td>
</tr>
<tr>
<td>QM1_fast</td>
<td>QM1 &amp; QM2</td>
<td>QM1_fast (at QM2)</td>
<td>QM1_fast_from_QM2</td>
</tr>
<tr>
<td>QM1_relief</td>
<td>QM1 &amp; QM2</td>
<td>QM1_relief (at QM2)</td>
<td>QM1_relief_from_QM2</td>
</tr>
<tr>
<td>QM1_relief</td>
<td>QM2 &amp; QM3</td>
<td>QM1_relief (at QM3)</td>
<td>QM1_relief_from_QM3</td>
</tr>
<tr>
<td>QM2</td>
<td>QM1 &amp; QM2</td>
<td>QM2 (at QM1)</td>
<td>QM2_from_QM1</td>
</tr>
<tr>
<td>QM2_fast</td>
<td>QM1 &amp; QM2</td>
<td>QM2_fast (at QM1)</td>
<td>QM2_fast_from_QM1</td>
</tr>
<tr>
<td>QM3</td>
<td>QM1 &amp; QM2</td>
<td>QM3 (at QM1)</td>
<td>QM3_from_QM1</td>
</tr>
<tr>
<td>QM3</td>
<td>QM2 &amp; QM3</td>
<td>QM3 (at QM2)</td>
<td>QM3_from_QM2</td>
</tr>
<tr>
<td>QM3_relief</td>
<td>QM1 &amp; QM2</td>
<td>QM3_relief (at QM1)</td>
<td>QM3_relief_from_QM1</td>
</tr>
<tr>
<td>QM3_relief</td>
<td>QM2 &amp; QM3</td>
<td>QM3_relief (at QM2)</td>
<td>QM3_relief_from_QM2</td>
</tr>
</tbody>
</table>

Notes:
1. On WebSphere MQ for z/OS, queue manager names are limited to 4 characters.
2. You are strongly recommended to name all the channels in your network uniquely. As shown in Table 1 on page 30, including the source and target queue manager names in the channel name is a good way to do this.

**Network planner**

This chapter has discussed application designer, systems administrator, and channel planner functions. Creating a network assumes that there is another, higher level function of *network planner* whose plans are implemented by the other members of the team.

If an application is used widely, it is more economical to think in terms of local access sites for the concentration of message traffic, using wide-band links between the local access sites, as shown in Figure 18.

In this example there are two main systems and a number of satellite systems (The actual configuration would depend on business considerations.) There are two concentrator queue managers located at convenient centers. Each QM-concentrator has message channels to the local queue managers:

- QM-concentrator 1 has message channels to each of the three local queue managers, QM1, QM2, and QM3. The applications using these queue managers can communicate with each other through the QM-concentrators.
- QM-concentrator 2 has message channels to each of the three local queue managers, QM4, QM5, and QM6. The applications using these queue managers can communicate with each other through the QM-concentrators.
- The QM-concentrators have message channels between themselves thus allowing any application at a queue manager to exchange messages with any other application at another queue manager.
Figure 18. Network diagram showing QM-concentrators
## Part 2. How intercommunication works

### Chapter 4. WebSphere MQ distributed-messaging techniques

- Message flow control ........................................ 35  
  - Queue names in transmission header .................. 36  
  - How to create queue manager and reply-to aliases .......... 36  
- Putting messages on remote queues ..................... 37  
- More about name resolution ................................. 38  
- Choosing the transmission queue ....................... 39  
- Receiving messages ....................................... 40  
  - Receiving alias queue manager names .................. 40  
- Passing messages through your system ................... 41  
  - Method 1: Using the incoming location name ........ 41  
  - Method 2: Using an alias for the queue manager .......... 42  
  - Method 3: Selecting a transmission queue ............. 42  
- Separating message flows ................................ 42  
- Concentrating messages to diverse locations ........... 44  
- Diverting message flows to another destination ......... 45  
- Sending messages to a distribution list ............... 46  
- Reply-to queue ........................................... 47  
  - Reply-to queue alias example ............................ 48  
  - Definitions used in this example at QM1 ............. 50  
  - Definitions used in this example at QM2 ............. 50  
  - Put definition at QM1 .................................. 50  
  - Put definition at QM2 .................................. 50  
- How the example works .................................... 50  
- How the queue manager makes use of the reply-to queue alias .................... 51  
- Networking considerations ................................. 52  
  - Networking considerations walk-through ............. 53  
- Managing queue name translations ....................... 53  
- Channel message sequence numbering ................... 54  
- Sequence retrieval of messages .......................... 55  
- Sequence of retrieval of fast, nonpersistent messages .......... 55  
- Loopback testing ........................................... 56

### Chapter 5. DQM implementation

- Functions of DQM .......................................... 57  
- Message sending and receiving ............................ 58  
- Channel parameters ...................................... 59  
- Channel status and sequence numbers ................... 59  
- Channel control function ................................ 59  
- Preparing channels ...................................... 60  
  - Auto-definition of receiver and server-connection channels ........ 60  
  - Defining other objects ................................ 61  
  - Multiple message channels per transmission queue ........ 61  
- Starting a channel ....................................... 61  
- Channel states ........................................... 61  
- Current and active ...................................... 62  
- Channel errors ........................................... 65

### Chapter 6. Channel attributes

- Channel attributes ........................................ 77  
- Channel attributes in alphabetical order .................. 77  
  - Alter date (Altdate) .................................. 80  
  - Alter time (AlttimE) .................................. 80  
  - Auto start (Autostart) ................................ 80  
  - Batch heartbeat interval (Batchhb) .................... 80  
  - Batch interval (BatchInt) ............................... 81  
  - Batch size (Batchsz) ................................ 81  
  - Channel name (Channel) ................................. 82  
  - Channel type (Chltype) ................................. 83  
  - Cics profile name ..................................... 83  
  - Cluster (Cluster) ....................................... 83  
  - Cluster namelist (Clusnl) ............................... 83  
  - Connection name (Conname) ............................ 84  
  - Convert message (Convert) ............................. 85  
  - Description (Descr) ................................... 86  
  - Disconnect interval (Discont) ......................... 86  
  - Heartbeat interval (HbInt) ............................. 87  
  - KeepAlive interval (Kaint) ............................. 87  
  - Local address (Locladdr) ............................... 88  
  - Long retry count (Lortcnt) ............................ 88  
  - Long retry interval (Lortint) ......................... 89  
  - LU 6.2 mode name (Modename) ........................ 89  
  - LU 6.2 transaction program name (Tpname) ............ 89  
  - Maximum message length (Maxmsgl) .................... 90  
  - Maximum transmission size ............................ 90  
  - Message channel agent name (Mcaname) ............... 91  
  - Message channel agent type (Mctype) ................. 91  
  - Message channel agent user identifier (MCuserid) .......... 91  
  - Message exit name (Msgexit) .......................... 92
## Intercommunication

<table>
<thead>
<tr>
<th>Configuration Item</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Message exit user data (MSGDATA)</td>
<td>92</td>
</tr>
<tr>
<td>Message-retry exit name (MREXIT)</td>
<td>92</td>
</tr>
<tr>
<td>Message-retry exit user data (MRDATA)</td>
<td>92</td>
</tr>
<tr>
<td>Message retry count (MRRTY)</td>
<td>92</td>
</tr>
<tr>
<td>Message retry interval (MRTMR)</td>
<td>93</td>
</tr>
<tr>
<td>Network-connection priority (NETPRTY)</td>
<td>93</td>
</tr>
<tr>
<td>Nonpersistent message speed (NPMSPED)</td>
<td>93</td>
</tr>
<tr>
<td>Password (PASSWORD)</td>
<td>93</td>
</tr>
<tr>
<td>PUT authority (PUTAUT)</td>
<td>94</td>
</tr>
<tr>
<td>Queue manager name (QMNAME)</td>
<td>95</td>
</tr>
<tr>
<td>Receive exit name (RCVEXIT)</td>
<td>95</td>
</tr>
<tr>
<td>Receive exit user data (RCVDATA)</td>
<td>95</td>
</tr>
<tr>
<td>Security exit name (SCYEXIT)</td>
<td>96</td>
</tr>
<tr>
<td>Security exit user data (SCYDATA)</td>
<td>96</td>
</tr>
<tr>
<td>Send exit name (SENDEXIT)</td>
<td>96</td>
</tr>
<tr>
<td>Send exit user data (SENDDATA)</td>
<td>96</td>
</tr>
<tr>
<td>Sequence number wrap (SEQWRAP)</td>
<td>96</td>
</tr>
<tr>
<td>Sequential delivery</td>
<td>97</td>
</tr>
<tr>
<td>Short retry count (SHORTRTY)</td>
<td>97</td>
</tr>
<tr>
<td>Short retry interval (SHORTTMR)</td>
<td>97</td>
</tr>
<tr>
<td>SSL Cipher Specification (SSLCIPH)</td>
<td>98</td>
</tr>
<tr>
<td>SSL Client Authentication (SSLCAUTH)</td>
<td>98</td>
</tr>
<tr>
<td>SSL Peer (SSLPEER)</td>
<td>98</td>
</tr>
<tr>
<td>Target system identifier</td>
<td>99</td>
</tr>
<tr>
<td>Transaction identifier</td>
<td>99</td>
</tr>
<tr>
<td>Transmission queue name (XMITQ)</td>
<td>99</td>
</tr>
<tr>
<td>Transport type (TRPTYPE)</td>
<td>100</td>
</tr>
<tr>
<td>User ID (USERID)</td>
<td>100</td>
</tr>
</tbody>
</table>

### Chapter 7. Example configuration chapters in this book

<table>
<thead>
<tr>
<th>Configuration Item</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Network infrastructure</td>
<td>101</td>
</tr>
<tr>
<td>Communications software</td>
<td>102</td>
</tr>
<tr>
<td>How to use the communication examples</td>
<td>103</td>
</tr>
<tr>
<td>IT responsibilities</td>
<td>104</td>
</tr>
</tbody>
</table>
Chapter 4. WebSphere MQ distributed-messaging techniques

This chapter describes techniques that are of use when planning channels. It introduces the concept of message flow control and explains how this is arranged in distributed queue management (DQM). It gives more detailed information about the concepts introduced in the preceding chapters and starts to show how you might use distributed queue management. This chapter covers the following topics:

- “Message flow control” on page 35
- “Putting messages on remote queues” on page 37
- “Choosing the transmission queue” on page 39
- “Receiving messages” on page 40
- “Passing messages through your system” on page 41
- “Choosing the transmission queue” on page 42
- “Concentrating messages to diverse locations” on page 44
- “Sending messages to a distribution list” on page 46
- “Reply-to queue” on page 47
- “Networking considerations” on page 52
- “Return routing” on page 53
- “Managing queue name translations” on page 53
- “Channel message sequence numbering” on page 54
- “Loopback testing” on page 56

Message flow control

Message flow control is a task that involves the setting up and maintenance of message routes between queue managers. This is very important for routes that multi-hop through many queue managers.

You control message flow using a number of techniques that were introduced in Chapter 2, “Making your applications communicate” on page 17. If your queue manager is in a cluster, message flow is controlled using different techniques, as described in the WebSphere MQ Queue Manager Clusters book. If your queue managers are in a queue sharing group and intra-group queuing (IGQ) is enabled, then the message flow can be controlled by IGQ agents, which are described in Chapter 37, “Intra-group queuing” on page 461.

This chapter describes how you use your system’s queues, alias queue definitions, and message channels to achieve message flow control.

You make use of the following objects:

- Transmission queues
- Message channels
- Remote queue definition
- Queue manager alias definition
- Reply-to queue alias definition

The queue manager and queue objects are described in the WebSphere MQ System Administration Guide book for WebSphere MQ on AIX, HP-UX, Linux, Solaris, and Windows systems, and MQSeries V5.1 for Compaq Tru64 UNIX, and OS/2 Warp, in the WebSphere MQ for iSeries V5.3 System Administration book for WebSphere MQ for iSeries, in the WebSphere MQ for z/OS Concepts and Planning Guide for WebSphere MQ for z/OS, or in the MQSeries System Management Guide for your
platform. Message channels are described in “Message channels” on page 8. The following techniques use these objects to create message flows in your system:

- Putting messages to remote queues
- Routing via particular transmission queues
- Receiving messages
- Passing messages through your system
- Separating message flows
- Switching a message flow to another destination
- Resolving the reply-to queue name to an alias name

Note

All the concepts described in this chapter are relevant for all nodes in a network, and include sending and receiving ends of message channels. For this reason, only one node is illustrated in most examples, except where the example requires explicit cooperation by the administrator at the other end of a message channel.

Before proceeding to the individual techniques it is useful to recap on the concepts of name resolution and the three ways of using remote queue definitions. See Chapter 3, “More about intercommunication” on page 25.

Queue names in transmission header

The queue name used by the application, the logical queue name, is resolved by the queue manager to the destination queue name, that is, the physical queue name. This destination queue name travels with the message in a separate data area, the transmission header, until the destination queue has been reached after which the transmission header is stripped off.

You will be changing the queue manager part of this queue name when you create parallel classes of service. Remember to return the queue manager name to the original name when the end of the class of service diversion has been reached.

How to create queue manager and reply-to aliases

As discussed above, the remote queue definition object is used in three different ways. Table 2 on page 37 explains how to define each of the three ways:

- Using a remote queue definition to redefine a local queue name.
  The application provides only the queue name when opening a queue, and this queue name is the name of the remote queue definition.
  The remote queue definition contains the names of the target queue and queue manager, and optionally, the definition can contain the name of the transmission queue to be used. If no transmission queue name is provided, the queue manager uses the queue manager name, taken from the remote queue definition, for the transmission queue name. If a transmission queue of this name is not defined, but a default transmission queue is defined, the default transmission queue is used.

- Using a remote queue definition to redefine a queue manager name.
  The application, or channel program, provides a queue name together with the remote queue manager name when opening the queue.
  If you have provided a remote queue definition with the same name as the queue manager name, and you have left the queue name in the definition blank, then the queue manager will substitute the queue manager name in the open call with the queue manager name in the definition.
In addition, the definition can contain the name of the transmission queue to be used. If no transmission queue name is provided, the queue manager takes the queue manager name, taken from the remote queue definition, for the transmission queue name. If a transmission queue of this name is not defined, but a default transmission queue is defined, the default transmission queue is used.

- Using a remote queue definition to redefine a reply-to queue name.
  Each time an application puts a message to a queue, it may provide the name of a reply-to queue for answer messages but with the queue manager name blank. If you provide a remote queue definition with the same name as the reply-to queue then the local queue manager replaces the reply-to queue name with the queue name from your definition.
  You may provide a queue manager name in the definition, but not a transmission queue name.

<table>
<thead>
<tr>
<th>Usage</th>
<th>Queue manager name</th>
<th>Queue name</th>
<th>Transmission queue name</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Remote queue definition (on OPEN call)</td>
<td>blank or local QM</td>
<td>(*) required</td>
<td>-</td>
</tr>
<tr>
<td>Supplied in the call</td>
<td>required</td>
<td>required</td>
<td>optional</td>
</tr>
<tr>
<td>Supplied in the definition</td>
<td>(*) required and not local QM</td>
<td>required</td>
<td>-</td>
</tr>
<tr>
<td>2. Queue manager alias (on OPEN call)</td>
<td>(*) required and not local QM</td>
<td>required</td>
<td>-</td>
</tr>
<tr>
<td>Supplied in the call</td>
<td>required</td>
<td>blank</td>
<td>optional</td>
</tr>
<tr>
<td>Supplied in the definition</td>
<td>blank</td>
<td>(*) required</td>
<td>-</td>
</tr>
<tr>
<td>3. Reply-to queue alias (on PUT call)</td>
<td>blank</td>
<td>(*) required</td>
<td>-</td>
</tr>
<tr>
<td>Supplied in the call</td>
<td>optional</td>
<td>optional</td>
<td>blank</td>
</tr>
</tbody>
</table>

Note: (*) means that this name is the name of the definition object

For a formal description, see Appendix C, "Queue name resolution" on page 711

Putting messages on remote queues

In a distributed-queuing environment, a transmission queue and channel are the focal point for all messages to a location whether the messages originate from applications in your local system, or arrive through channels from an adjacent system. This is shown in Figure 19 on page 38 where an application is placing messages on a logical queue named ‘QA_norm’. The name resolution uses the remote queue definition ‘QA_norm’ to select the transmission queue ‘QMB’, and adds a transmission header to the messages stating ‘QA_norm at QMB’.

Messages arriving from the adjacent system on ‘Channel_back’ have a transmission header with the physical queue name ‘QA_norm at QMB’, for example. These messages are placed unchanged on transmission queue QMB.

The channel moves the messages to an adjacent queue manager.
If you are the WebSphere MQ system administrator, you must:

- Define the message channel from the adjacent system
- Define the message channel to the adjacent system
- Create the transmission queue ‘QMB’
- Define the remote queue object ‘QA_norm’ to resolve the queue name used by applications to the desired destination queue name, destination queue manager name, and transmission queue name

In a clustering environment, you only need to define a cluster-receiver channel at the local queue manager. You do not need to define a transmission queue or a remote queue object. For information about this, see the WebSphere MQ Queue Manager Clusters book.

**More about name resolution**

The effect of the remote queue definition is to define a physical destination queue name and queue manager name; these names are put in the transmission headers of messages.

Incoming messages from an adjacent system have already had this type of name resolution carried out by the original queue manager, and have the transmission header showing the physical destination queue name and queue manager name. These messages are unaffected by remote queue definitions.
Choosing the transmission queue

Choosing the transmission queue

In a distributed-queuing environment, when you need to change a message flow from one channel to another, use the same system configuration as shown in Figure 19 on page 38. Figure 20 shows how you use the remote queue definition to send messages over a different transmission queue, and therefore over a different channel, to the same adjacent queue manager.

For the configuration shown in Figure 20 you must provide:

- The remote queue object 'QA_norm' to choose:
  - Queue 'QA_norm' at the remote queue manager
  - Transmission queue 'TX1'
  - Queue manager 'QMB_priority'

- The transmission queue 'TX1'. Specify this in the definition of the channel to the adjacent system

Messages are placed on transmission queue 'TX1' with a transmission header containing 'QA_norm at QMB_priority', and are sent over the channel to the adjacent system.

The channel_back has been left out of this illustration because it would need a queue manager alias; this is discussed in the following example.

In a clustering environment, you do not need to define a transmission queue or a remote queue definition. For more information about this, see the WebSphere MQ Queue Manager Clusters book.
Receiving messages

As well as arranging for messages to be sent, the system administrator must also arrange for messages to be received from adjacent queue managers. Received messages contain the physical name of the destination queue manager and queue in the transmission header. They are treated exactly the same as messages from a local application that specifies both queue manager name and queue name. Because of this, you need to ensure that messages entering your system do not have an unintentional name resolution carried out. See Figure 21 for this scenario.

For this configuration, you must prepare:

- Message channels to receive messages from adjacent queue managers
- A queue manager alias definition to resolve an incoming message flow, ‘QMB_priority’, to the local queue manager name, ‘QMB’
- The local queue, ‘QA_norm’, if it does not already exist

**Receiving alias queue manager names**

The use of the queue manager alias definition in this illustration has not selected a different destination queue manager. Messages passing through this local queue manager and addressed to ‘QMB_priority’ are intended for queue manager ‘QMB’. The alias queue manager name is used to create the separate message flow.
Passing messages through your system

Following on from the technique shown in Figure 21 on page 40, where you saw how an alias flow is captured, Figure 22 illustrates the ways networks are built up by bringing together the techniques we have discussed.

The configuration shows a channel delivering three messages with different destinations:
1. ‘QB at QMC’
2. ‘QB at QMD_norm’
3. ‘QB at QMD_PRIORITY’

You must pass the first message flow through your system unchanged; the second message flow through a different transmission queue and channel, while reverting the messages from the alias queue manager name ‘QMD_norm’ to the physical location ‘QMD’; and the third message flow simply chooses a different transmission queue without any other change.

In a clustering environment, all messages are passed through the cluster transmission queue, SYSTEM.CLUSTER.TRANSMIT.QUEUE. This is illustrated in Figure 4 on page 7.

The following methods describe techniques applicable to a distributed-queuing environment:

**Method 1: Using the incoming location name**

When you are going to receive messages with a transmission header containing another location name, the simplest preparation is to have a transmission queue
Passing messages through system

with that name, ‘QMC’ in this example, as a part of a channel to an adjacent queue manager. The messages are delivered unchanged.

Method 2: Using an alias for the queue manager

The second method is to use the queue manager alias object definition, but specify a new location name, ‘QMD’, as well as a particular transmission queue, ‘TX1’.

This action:
- Terminates the alias message flow set up by the queue manager name alias ‘QMD_norm’. That is the named class of service ‘QMD_norm’.
- Changes the transmission headers on these messages from ‘QMD_norm’ to ‘QMD’.

Method 3: Selecting a transmission queue

The third method is to have a queue manager alias object defined with the same name as the destination location, ‘QMD_PRIORITY’, and use the definition to select a particular transmission queue, ‘QMD_fast’, and therefore another channel. The transmission headers on these messages remain unchanged.

Using these methods

For these configurations, you must prepare the:
- Input channel definitions
- Output channel definitions
- Transmission queues:
  - QMC
  - TX1
  - QMD_fast
- Queue manager alias definitions:
  - QMD_norm with ‘QMD_norm to QMD via TX1’
  - QMD_PRIORITY with ‘QMD_PRIORITY to QMD_PRIORITY via QMD_fast’

Note

None of the message flows shown in the example changes the destination queue. The queue manager name aliases simply provide separation of message flows.

Separating message flows

In a distributed-queuing environment, the need to separate messages to the same queue manager into different message flows can arise for a number of reasons. For example:
- You may need to provide a separate flow for very large, medium, and small messages. This also applies in a clustering environment and, in this case, you may create clusters that overlap. There are a number of reasons you might do this, for example:
  - To allow different organizations to have their own administration.
  - To allow independent applications to be administered separately.
  - To create a class of service. For example you could have a cluster called STAFF that is a subset of the cluster called STUDENTS. When you put a message to a queue advertised in the STAFF cluster, a restricted channel is
used. When you put a message to a queue advertised in the STUDENTS cluster, either a general channel or a restricted channel may be used.

- To create test and production environments.
- It may be necessary to route incoming messages via different paths from the path of the locally generated messages.
- Your installation may require to schedule the movement of messages at certain times (for example, overnight) and the messages then need to be stored in reserved queues until scheduled.

In the example shown in Figure 23, the two incoming flows are to alias queue manager names ‘QMC_small’ and ‘QMC_large’. You provide these flows with a queue manager alias definition to capture these flows for the local queue manager. You have an application addressing two remote queues and you need these message flows to be kept separate. You provide two remote queue definitions that specify the same location, ‘QMC’, but specify different transmission queues. This keeps the flows separate, and nothing extra is needed at the far end as they have the same destination queue manager name in the transmission headers. You provide:

- The incoming channel definitions
- The two remote queue definitions QB_small and QB_large
- The two queue manager alias definitions QMC_small and QMC_large
- The three sending channel definitions
- Three transmission queues: TX_small, TX_large, and TX_external

Figure 23. Separating messages flows
Coordination with adjacent systems

When you use a queue manager alias to create a separate message flow, you need to coordinate this activity with the system administrator at the remote end of the message channel to ensure that the corresponding queue manager alias is available there.

Concentrating messages to diverse locations

Figure 24 illustrates a distributed-queuing technique for concentrating messages that are destined for various locations on to one channel. Two possible uses would be:

- Concentrating message traffic through a gateway
- Using wide bandwidth highways between nodes

In this example, messages from different sources, local and adjacent, and having different destination queues and queue managers, are flowed via transmission queue ‘TX1’ to queue manager QMC. Queue manager QMC delivers the messages according to the destinations, one set to a transmission queue ‘QMD’ for onward
transmission to queue manager QMD, another set to a transmission queue ‘QME’ for onward transmission to queue manager QME, while other messages are put on the local queue ‘QA’.

You must provide:
• Channel definitions
• Transmission queue TX1
• Remote queue definitions:
  – QA with ‘QA at QMC via TX1’
  – QB with ‘QB at QMD via TX1’
• Queue manager alias definition:
  – QME with ‘QME via TX1’

The complementary administrator who is configuring QMC must provide:
• Receiving channel definition with the same channel name
• Transmission queue QMD with associated sending channel definition
• Transmission queue QME with associated sending channel definition
• Local queue object QA.

### Diverting message flows to another destination

*Figure 25. Diverting message streams to another destination*

`Figure 25` illustrates how you can redefine the destination of certain messages. Incoming messages to QMA are destined for ‘QB at QMC’. They would normally arrive at QMA and be placed on a transmission queue called QMC which would have been part of a channel to QMC. QMA must divert the messages to QMD, but is able to reach QMD only over QMB. This method is useful when you need to move a service from one location to another, and allow subscribers to continue to send messages on a temporary basis until they have adjusted to the new address.

The method of rerouting incoming messages destined for a certain queue manager to a different queue manager uses:
• A queue manager alias to change the destination queue manager to another queue manager, and to select a transmission queue to the adjacent system
• A transmission queue to serve the adjacent queue manager
• A transmission queue at the adjacent queue manager for onward routing to the destination queue manager

You must provide:
• Channel_back definition
Diverting message flows

- Queue manager alias object definition QMC with QB at QMD via QMB
- Channel_out definition
- The associated transmission queue QMB

The complementary administrator who is configuring QMB must provide:
- The corresponding channel_back definition
- The transmission queue, QMD
- The associated channel definition to QMD

You can use aliases within a clustering environment. For information about this, see the WebSphere MQ Queue Manager Clusters book.

Sending messages to a distribution list

In WebSphere MQ for AIX, iSeries, HP-UX, Linux, Solaris, and Windows systems, and MQSeries V5.1 for OS/2 Warp, an application can send a message to several destinations with a single MQPUT call. This applies in both a distributed-queuing environment and a clustering environment. You have to define the destinations in a distribution list, as described in the WebSphere MQ Application Programming Guide book.

Not all queue managers support distribution lists. When an MCA establishes a connection with a partner, it determines whether or not the partner supports distribution lists and sets a flag on the transmission queue accordingly. If an application tries to send a message that is destined for a distribution list but the partner does not support distribution lists, the sending MCA intercepts the message and puts it onto the transmission queue once for each intended destination.

A receiving MCA ensures that messages sent to a distribution list are safely received at all the intended destinations. If any destinations fail, the MCA establishes which ones have failed so that it can generate exception reports for them and can try to re-send the messages to them.
A complete remote queue processing loop using a reply-to queue is shown in Figure 26. This applies in both a distributed-queuing environment and a clustering environment. The details are as shown in Table 6 on page 54.

The application opens QA at QMB and puts messages on that queue. The messages are given a reply-to queue name of QR, without the queue manager name being specified. Queue manager QMA finds the reply-to queue object QR and extracts from it the alias name of QRR and the queue manager name QMA_class1. These names are put into the reply-to fields of the messages.

Reply messages from applications at QMB are addressed to QRR at QMA_class1. The queue manager alias name definition QMA_class1 is used by the queue manager to flow the messages to itself, and to queue QRR.

This scenario depicts the way you give applications the facility to choose a class of service for reply messages, the class being implemented by the transmission queue QMA_class1 at QMB, together with the queue manager alias definition, QMA_class1 at QMA. In this way, you can change an application’s reply-to queue so that the flows are segregated without involving the application. That is, the application always chooses QR for this particular class of service, and you have the opportunity to change the class of service with the reply-to queue definition QR.

You must create:
- Reply-to queue definition QR
- Transmission queue object QMB
- Channel_out definition
- Channel_back definition
- Queue manager alias definition QMA_class1
- Local queue object QRR, if it does not exist

The complementary administrator at the adjacent system must create:
- Receiving channel definition
Reply-to queue

- Transmission queue object QMA_class1
- Associated sending channel
- Local queue object QA.

Your application programs use:
- Reply-to queue name QR in put calls
- Queue name QRR in get calls

In this way, you may change the class of service as necessary, without involving the application, by changing the reply-to alias ‘QR’, together with the transmission queue ‘QMA_class1’ and queue manager alias ‘QMA_class1’.

If no reply-to alias object is found when the message is put on the queue, the local queue manager name is inserted in the blank reply-to queue manager name field, and the reply-to queue name remains unchanged.

Name resolution restriction
Because the name resolution has been carried out for the reply-to queue at ‘QMA’ when the original message was put, no further name resolution is allowed at ‘QMB’, that is, the message is put with the physical name of the reply-to queue by the replying application.

Note that the applications must be aware of the naming convention that the name they use for the reply-to queue is different from the name of the actual queue where the return messages are to be found.

For example, when two classes of service are provided for the use of applications with reply-to queue alias names of ‘C1_alias’, and ‘C2_alias’, the applications use these names as reply-to queue names in the message put calls, but the applications will actually expect messages to appear in queues ‘C1’ and ‘C2’, respectively.

However, an application is able to make an inquiry call on the reply-to alias queue to check for itself the name of the real queue it must use to get the reply messages.

Reply-to queue alias example
This example illustrates the use of a reply-to alias to select a different route (transmission queue) for returned messages. The use of this facility requires the reply-to queue name to be changed in cooperation with the applications.

As shown in Figure 27 on page 49, the return route must be available for the reply messages, including the transmission queue, channel, and queue manager alias.
This example is for requester applications at ‘QM1’ that send messages to server applications at ‘QM2’. The servers’ messages are to be returned through an alternative channel using transmission queue ‘QM1_relief’ (the default return channel would be served with a transmission queue ‘QM1’).

The reply-to queue alias is a particular use of the remote queue definition named ‘Answer_alias’. Applications at QM1 include this name, ‘Answer_alias’, in the reply-to field of all messages that they put on queue ‘Inquiry’.

Reply-to queue definition ‘Answer_alias’ is defined as ‘Answer at QM1_relief’. Applications at QM1 expect their replies to appear in the local queue named ‘Answer’.

Server applications at QM2 use the reply-to field of received messages to obtain the queue and queue manager names for the reply messages to the requester at QM1.

**Definitions used in this example at QM1**
The WebSphere MQ system administrator at QM1 must ensure that the reply-to queue ‘Answer’ is created along with the other objects. The name of the queue manager alias, marked with a ‘*’, must agree with the queue manager name in the reply-to queue alias definition, also marked with an ‘*’.
Reply-to queue

<table>
<thead>
<tr>
<th>Object Definition</th>
<th>Local transmission queue</th>
<th>Query</th>
<th>Remote queue manager name</th>
<th>Inquiry</th>
<th>Remote queue name</th>
<th>Inquiry</th>
<th>Transmission queue name</th>
<th>Query</th>
<th>QM2 (DEFAULT)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Queue manager alias</td>
<td>Object name</td>
<td>Query</td>
<td>Queue manager name</td>
<td>Query</td>
<td>QM1_relief *</td>
<td></td>
<td>Queue name</td>
<td>(blank)</td>
<td></td>
</tr>
<tr>
<td>Reply-to queue alias</td>
<td>Object name</td>
<td>Answer_alias</td>
<td>Remote queue manager name</td>
<td>QM1_relief *</td>
<td></td>
<td></td>
<td>Remote queue name</td>
<td>Answer</td>
<td></td>
</tr>
</tbody>
</table>

Definitions used in this example at QM2

The WebSphere MQ system administrator at QM2 must ensure that the local queue exists for the incoming messages, and that the correctly named transmission queue is available for the reply messages.

<table>
<thead>
<tr>
<th>Object Definition</th>
<th>Local queue</th>
<th>Inquiry</th>
<th>Transmission queue</th>
<th>QM1_relief</th>
</tr>
</thead>
</table>

Put definition at QM1

Applications fill the reply-to fields with the reply-to queue alias name, and leave the queue manager name field blank.

<table>
<thead>
<tr>
<th>Field</th>
<th>Content</th>
</tr>
</thead>
<tbody>
<tr>
<td>Queue name</td>
<td>Inquiry</td>
</tr>
<tr>
<td>Queue manager name</td>
<td>(blank)</td>
</tr>
<tr>
<td>Reply-to queue name</td>
<td>Answer_alias</td>
</tr>
<tr>
<td>Reply-to queue manager</td>
<td>(blank)</td>
</tr>
</tbody>
</table>

Put definition at QM2

Applications at QM2 retrieve the reply-to queue name and queue manager name from the original message and use them when putting the reply message on the reply-to queue.

<table>
<thead>
<tr>
<th>Field</th>
<th>Content</th>
</tr>
</thead>
<tbody>
<tr>
<td>Queue name</td>
<td>Answer</td>
</tr>
<tr>
<td>Queue manager name</td>
<td>QM1_relief</td>
</tr>
</tbody>
</table>

How the example works

In this example, requester applications at QM1 always use ‘Answer_alias’ as their reply-to queue in the relevant field of the put call, and they always retrieve their messages from the queue named ‘Answer’.

The reply-to queue alias definitions are available for use by the QM1 system administrator to change the name of the reply-to queue ‘Answer’, and of the return route ‘QM1_relief’.

Changing the queue name ‘Answer’ is normally not useful because the QM1 applications are expecting their answers in this queue. However, the QM1 system administrator is able to change the return route (class of service), as necessary.
How the queue manager makes use of the reply-to queue alias

Queue manager QM1 retrieves the definitions from the reply-to queue alias when the reply-to queue name, included in the put call by the application, is the same as the reply-to queue alias, and the queue manager part is blank.

The queue manager replaces the reply-to queue name in the put call with the queue name from the definition. It replaces the blank queue manager name in the put call with the queue manager name from the definition.

These names are carried with the message in the message descriptor.

<table>
<thead>
<tr>
<th>Field name</th>
<th>Put call</th>
<th>Transmission header</th>
</tr>
</thead>
<tbody>
<tr>
<td>Queue name</td>
<td>Answer_alias</td>
<td>Answer</td>
</tr>
<tr>
<td>Queue manager name</td>
<td>(blank)</td>
<td>QM1_relief</td>
</tr>
</tbody>
</table>

Reply-to queue alias walk-through

To complete this example, let us take a walk through the process, from an application putting a message on a remote queue at queue manager ‘QM1’, through to the same application removing the reply message from the alias reply-to queue.

1. The application opens a queue named ‘Inquiry’, and puts messages to it. The application sets the reply-to fields of the message descriptor to:
   - Reply-to queue name: Answerable
   - Reply-to queue manager name: (blank)

2. Queue manager ‘QM1’ responds to the blank queue manager name by checking for a remote queue definition with the name ‘Answer_alias’. If none is found, the queue manager places its own name, ‘QM1’, in the reply-to queue manager field of the message descriptor.

3. If the queue manager finds a remote queue definition with the name ‘Answer_alias’, it extracts the queue name and queue manager names from the definition (queue name=‘Answer’ and queue manager name=’QM1_relief’) and puts them into the reply-to fields of the message descriptor.

4. The queue manager ‘QM1’ uses the remote queue definition ‘Inquiry’ to determine that the intended destination queue is at queue manager ‘QM2’, and the message is placed on the transmission queue ‘QM2’. ‘QM2’ is the default transmission queue name for messages destined for queues at queue manager ‘QM2’.

5. When queue manager ‘QM1’ puts the message on the transmission queue, it adds a transmission header to the message. This header contains the name of the destination queue, ‘Inquiry’, and the destination queue manager, ‘QM2’.

6. The message arrives at queue manager ‘QM2’, and is placed on the ‘Inquiry’ local queue.

7. An application gets the message from this queue and processes the message. The application prepares a reply message, and puts this reply message on the reply-to queue name from the message descriptor of the original message. This is:
   - Reply-to queue name: Answer
   - Reply-to queue manager name: QM1_relief
8. Queue manager ‘QM2’ carries out the put command, and finding that the queue manager name, ‘QM1_relief’, is a remote queue manager, it places the message on the transmission queue with the same name, ‘QM1_relief’. The message is given a transmission header containing the name of the destination queue, ‘Answer’, and the destination queue manager, ‘QM1_relief’.

9. The message is transferred to queue manager ‘QM1’ where the queue manager, recognizing that the queue manager name ‘QM1_relief’ is an alias, extracts from the alias definition ‘QM1_relief’ the physical queue manager name ‘QM1’.

10. Queue manager ‘QM1’ then puts the message on the queue name contained in the transmission header, ‘Answer’.

11. The application extracts its reply message from the queue ‘Answer’.

**Networking considerations**

In a distributed-queuing environment, because message destinations are addressed with just a queue name and a queue manager name, the following rules apply:

1. Where the queue manager name is given, and the name is different from the local queue manager’s name:
   - A transmission queue must be available with the same name, and this transmission queue must be part of a message channel moving messages to another queue manager, or
   - A queue manager alias definition must exist to resolve the queue manager name to the same, or another queue manager name, and optional transmission queue, or
   - If the transmission queue name cannot be resolved, and a default transmission queue has been defined, the default transmission queue is used.

2. Where only the queue name is supplied, a queue of any type but with the same name must be available on the local queue manager. This queue may be a remote queue definition which resolves to: a transmission queue to an adjacent queue manager, a queue manager name, and an optional transmission queue.

To see how this works in a clustering environment, see the [WebSphere MQ Queue Manager Clusters](web) book.

If the queue managers are running in a queue sharing group (QSG) and intra-group queuing (IGQ) is enabled, the SYSTEM.QSG.TRANSMIT.QUEUE may be used. For more information, see [Chapter 37, "Intra-group queuing" on page 461](web).

Consider the scenario of a message channel moving messages from one queue manager to another in a distributed-queuing environment.

The messages being moved have originated from any other queue manager in the network, and some messages may arrive that have an unknown queue manager name as destination. This can occur when a queue manager name has changed or has been removed from the system, for example.

The channel program recognizes this situation when it cannot find a transmission queue for these messages, and places the messages on your undelivered-message (dead-letter) queue. It is your responsibility to look for these messages and arrange for them to be forwarded to the correct destination, or to return them to the originator, where this can be ascertained.
Exception reports are generated in these circumstances, if report messages were requested in the original message.

**Name resolution convention**

It is strongly recommended that name resolution that changes the identity of the destination queue, (that is, logical to physical name changing), should only occur once, and only at the originating queue manager.

Subsequent use of the various alias possibilities should be used only when separating and combining message flows.

**Return routing**

Messages may contain a return address in the form of the name of a queue and queue manager. This applies in both a distributed-queuing environment and a clustering environment. This address is normally specified by the application that creates the message, but may be modified by any application that subsequently handles the message, including user exit applications.

Irrespective of the source of this address, any application handling the message may choose to use this address for returning answer, status, or report messages to the originating application.

The way these response messages are routed is not different from the way the original message is routed. You need to be aware that the message flows you create to other queue managers will need corresponding return flows.

**Physical name conflicts**

The destination reply-to queue name has been resolved to a physical queue name at the original queue manager, and must not be resolved again at the responding queue manager.

This is a likely possibility for name conflict problems that can only be prevented by a network-wide agreement on physical and logical queue names.

**Managing queue name translations**

This description is mainly provided for application designers and channel planners concerned with an individual system that has message channels to adjacent systems. It takes a local view of channel planning and control.

When you create a queue manager alias definition or a remote queue definition, the name resolution is carried out for every message carrying that name, regardless of the source of the message. To oversee this situation, which may involve large numbers of queues in a queue manager network, you keep tables of:

- The names of source queues and of source queue managers with respect to resolved queue names, resolved queue manager names, and resolved transmission queue names, with method of resolution

- The names of source queues with respect to:
  - Resolved destination queue names
  - Resolved destination queue manager names
Managing queue name translations

- Transmission queues
- Message channel names
- Adjacent system names
- Reply-to queue names

**Note:** The use of the term **source** in this context refers to the queue name or the queue manager name provided by the application, or a channel program when opening a queue for putting messages.

An example of each of these tables is shown in Table 4, Table 5, and Table 6.

The names in these tables are derived from the examples in this chapter, and this table is not intended as a practical example of queue name resolution in one node.

**Table 4. Queue name resolution at queue manager QMA**

<table>
<thead>
<tr>
<th>Source queue specified when queue is opened</th>
<th>Source queue manager specified when queue is opened</th>
<th>Resolved queue name</th>
<th>Resolved queue manager name</th>
<th>Resolved transmission queue name</th>
<th>Resolution type</th>
</tr>
</thead>
<tbody>
<tr>
<td>QA_norm</td>
<td>-</td>
<td>QA_norm</td>
<td>QMB</td>
<td>QMB</td>
<td>Remote queue</td>
</tr>
<tr>
<td>(any)</td>
<td>QMB</td>
<td>-</td>
<td>QMB</td>
<td>-</td>
<td>(none)</td>
</tr>
<tr>
<td>QA_norm</td>
<td>QMB</td>
<td>QA_norm</td>
<td>QMB</td>
<td>TX1</td>
<td>Remote queue</td>
</tr>
<tr>
<td>QB</td>
<td>QMC</td>
<td>QB</td>
<td>QMD</td>
<td>QMB</td>
<td>Queue manager alias</td>
</tr>
</tbody>
</table>

**Table 5. Queue name resolution at queue manager QMB**

<table>
<thead>
<tr>
<th>Source queue specified when queue is opened</th>
<th>Source queue manager specified when queue is opened</th>
<th>Resolved queue name</th>
<th>Resolved queue manager name</th>
<th>Resolved transmission queue name</th>
<th>Resolution type</th>
</tr>
</thead>
<tbody>
<tr>
<td>QA_norm</td>
<td>-</td>
<td>QA_norm</td>
<td>QMB</td>
<td>QMB</td>
<td>(none)</td>
</tr>
<tr>
<td>QA_norm</td>
<td>QMB</td>
<td>QA_norm</td>
<td>QMB</td>
<td>-</td>
<td>(none)</td>
</tr>
<tr>
<td>QA_norm</td>
<td>QMB_PRIORITY</td>
<td>QA_norm</td>
<td>QMB</td>
<td>-</td>
<td>Queue manager alias</td>
</tr>
<tr>
<td>(any)</td>
<td>QMC</td>
<td>(any)</td>
<td>QMC</td>
<td>QMC</td>
<td>(none)</td>
</tr>
<tr>
<td>(any)</td>
<td>QMD_norm</td>
<td>(any)</td>
<td>QMD_norm</td>
<td>TX1</td>
<td>Queue manager alias</td>
</tr>
<tr>
<td>(any)</td>
<td>QMD_PRIORITY</td>
<td>(any)</td>
<td>QMD_PRIORITY</td>
<td>QMD_fast</td>
<td>Queue manager alias</td>
</tr>
<tr>
<td>(any)</td>
<td>QMC_small</td>
<td>(any)</td>
<td>QMC_small</td>
<td>TX_small</td>
<td>Queue manager alias</td>
</tr>
<tr>
<td>(any)</td>
<td>QMC_large</td>
<td>(any)</td>
<td>QMC_large</td>
<td>TX_external</td>
<td>Queue manager alias</td>
</tr>
<tr>
<td>QB_small</td>
<td>QMC</td>
<td>QB_small</td>
<td>QMC</td>
<td>TX_small</td>
<td>Remote queue</td>
</tr>
<tr>
<td>QB_large</td>
<td>QMC</td>
<td>QB_large</td>
<td>QMC</td>
<td>TX_large</td>
<td>Remote queue</td>
</tr>
<tr>
<td>QA</td>
<td>QMC</td>
<td>QA</td>
<td>QMC</td>
<td>TX1</td>
<td>Remote queue</td>
</tr>
<tr>
<td>QB</td>
<td>QMD</td>
<td>QB</td>
<td>QMD</td>
<td>TX1</td>
<td>Remote queue</td>
</tr>
</tbody>
</table>

**Table 6. Reply-to queue name translation at queue manager QMA**

<table>
<thead>
<tr>
<th>Application design</th>
<th>Queue name for messages</th>
<th>Reply-to alias definition</th>
<th>Redefined to</th>
</tr>
</thead>
<tbody>
<tr>
<td>Local QMGR</td>
<td>Queue name for messages</td>
<td>Reply-to alias name</td>
<td>Redefined to</td>
</tr>
<tr>
<td>QMA</td>
<td>QRR</td>
<td>QRR at QMA_class1</td>
<td></td>
</tr>
</tbody>
</table>

**Channel message sequence numbering**

The channel uses sequence numbers to assure that messages are delivered, delivered without duplication, and stored in the same order as they were taken from the transmission queue. The sequence number is generated at the sending end of the channel and is incremented by one before being used, which means that the current sequence number is the number of the last message sent. This information can be displayed using DISPLAY CHSTATUS (see WebSphere MQ Script...
The sequence number and an identifier called the LUWID are stored in persistent storage for the last message transferred in a batch. These values are used during channel start-up to ensure that both ends of the link agree on which messages have been transferred successfully.

**Note:** On z/OS, if you are using distributed queuing with CICS, and you do not use the sequential delivery option, then message sequence numbering is not used.

### Sequential retrieval of messages

If an application puts a sequence of messages to the same destination queue, those messages can be retrieved in sequence by a single application with a sequence of MQGET operations, if the following conditions are met:

- All of the put requests were done from the same application.
- All of the put requests were either from the same unit of work, or all the put requests were made outside of a unit of work.
- The messages all have the same priority.
- The messages all have the same persistence.
- For remote queuing, the configuration is such that there can only be one path from the application making the put request, through its queue manager, through intercommunication, to the destination queue manager and the target queue.
- The messages are not put to a dead-letter queue (for example, if a queue is temporarily full).
- The application getting the message does not deliberately change the order of retrieval, for example by specifying a particular MsgId or CorrelId or by using message priorities.
- Only one application is doing get operations to retrieve the messages from the destination queue. If this is not the case, these applications must be designed to get all the messages in each sequence put by a sending application.

**Note:** Messages from other tasks and units of work may be interspersed with the sequence, even where the sequence was put from within a single unit of work.

If these conditions cannot be met, and the order of messages on the target queue is important, then the application can be coded to use its own message sequence number as part of the message to assure the order of the messages.

### Sequence of retrieval of fast, nonpersistent messages

WebSphere MQ for AIX, iSeries, HP-UX, Linux, Solaris, Windows systems, and z/OS without CICS, and MQSeries V5.1 for OS/2 Warp, nonpersistent messages on a fast channel may overtake persistent messages on the same channel and so arrive out of sequence. The receiving MCA puts the nonpersistent messages on the destination queue immediately and makes them visible. Persistent messages are not made visible until the next syncpoint.
Loopback testing

Loopback testing is a technique on non-z/OS platforms that allows you to test a communications link without actually linking to another machine. You set up a connection between two queue managers as though they are on separate machines, but you test the connection by looping back to another process on the same machine. This means that you can test your communications code without requiring an active network.

The way you do this depends on which products and protocols you are using. For example the command to allow TCP/IP loopback testing on OS/2 without a network, is:

```
ifconfig lo ipaddress
```

On Windows systems, you can use the “loopback” adapter.

Refer to the documentation for the products you are using for more information.
Chapter 5. DQM implementation

This chapter describes the implementation of the concepts introduced in Chapter 2, "Making your applications communicate" on page 17.

Distributed queue management (DQM):

- Enables you to define and control communication channels between queue managers
- Provides you with a message channel service to move messages from a type of local queue, known as a transmission queue, to communication links on a local system, and from communication links to local queues at a destination queue manager
- Provides you with facilities for monitoring the operation of channels and diagnosing problems, using panels, commands, and programs

This chapter discusses:

- "Functions of DQM"
- "Message sending and receiving" on page 58
- "Channel control function" on page 59
- "What happens when a message cannot be delivered?" on page 71
- "Initialization and configuration files" on page 73
- "Data conversion" on page 75
- "Writing your own message channel agents" on page 75

Functions of DQM

Distributed queue management has these functions:

- Message sending and receiving
- Channel control
- Initialization file
- Data conversion
- Channel exits

Channel definitions associate channel names with transmission queues, communication link identifiers, and channel attributes. Channel definitions are implemented in different ways on different platforms. Message sending and receiving is controlled by programs known as message channel agents (MCAs), which use the channel definitions to start up and control communication.

The MCAs in turn are controlled by DQM itself. The structure is platform dependent, but typically includes listeners and trigger monitors, together with operator commands and panels.

A message channel is a one-way pipe for moving messages from one queue manager to another. Thus a message channel has two end-points, represented by a pair of MCAs. Each end-point has a definition of its end of the message channel. For example, one end would define a sender, the other end a receiver.
Functions of DQM

For details of how to define channels, see:

- Chapter 8, “Monitoring and controlling channels on distributed platforms” on page 109
- Chapter 23, “Monitoring and controlling channels on z/OS” on page 341
- Chapter 28, “Monitoring and controlling channels in z/OS with CICS” on page 383
- Chapter 39, “Monitoring and controlling channels in WebSphere MQ for iSeries” on page 485

For information about channel exits, see Chapter 45, “Channel-exit programs” on page 571.

Message sending and receiving

Figure 28 shows the relationships between entities when messages are transmitted, and shows the flow of control.

Notes:

1. There is one MCA per channel, depending on the platform. There may be one or more channel control functions for a given queue manager.
2. The implementation of MCAs and channel control functions is highly platform dependent; they may be programs or processes or threads, and they may be a single entity or many comprising several independent or linked parts.
3. All components marked with a star can use the MQI.

Channel parameters

An MCA receives its parameters in one of several ways:

- If started by a command, the channel name is passed in a data area. The MCA then reads the channel definition directly to obtain its attributes.

- For sender, and in some cases server channels, the MCA can be started automatically by the queue manager trigger. The channel name is retrieved from the trigger process definition, where applicable, and is passed to the MCA. The remaining processing is the same as that described above.

- If started remotely by a sender, server, requester, or client-connection, the channel name is passed in the initial data from the partner message channel agent. The MCA reads the channel definition directly to obtain its attributes.

Certain attributes not defined in the channel definition are also negotiable:

Split messages
If one end does not support this, split messages will not be sent.

Conversion capability
If one end cannot perform the necessary code page conversion or numeric encoding conversion when needed, the other end must handle it. If neither end supports it, when needed, the channel cannot start.

Distribution list support
If one end does not support distribution lists, the partner MCA sets a flag in its transmission queue so that it will know to intercept messages intended for multiple destinations.

Channel status and sequence numbers

Message channel agent programs keep records of the current sequence number and logical unit of work number for each channel, and of the general status of the channel. Some platforms allow you to display this status information to help you control channels.

Channel control function

The channel control function provides facilities for you to define, monitor, and control channels. Commands are issued through panels, programs, or from a command line to the channel control function. The panel interface also displays channel status and channel definition data.

Note: For the channel control function on WebSphere MQ for UNIX and Windows, and MQSeries for OS/2 Warp, Compaq OpenVMS Alpha, and Compaq NonStop Kernel, you can use Programmable Command Formats or those WebSphere MQ commands (MQSC) and control commands that are detailed in Chapter 8, “Monitoring and controlling channels on distributed platforms” on page 109.

The commands fall into the following groups:

- Channel administration
- Channel control
- Channel status monitoring

Channel administration commands deal with the definitions of the channels. They enable you to:
Channel control function

- Create a channel definition
- Copy a channel definition
- Alter a channel definition
- Delete a channel definition

Channel control commands manage the operation of the channels. They enable you to:
- Start a channel
- Stop a channel
- Re-synchronize with partner (in some implementations)
- Reset message sequence numbers
- Resolve an in-doubt batch of messages
- Ping; send a test communication across the channel

Channel monitoring displays the state of channels, for example:
- Current channel settings
- Whether the channel is active or inactive
- Whether the channel terminated in a synchronized state

Preparing channels

Before trying to start a message channel or MQI channel, you must make sure that all the attributes of the local and remote channel definitions are correct and compatible. Chapter 6, “Channel attributes” on page 77 describes the channel definitions and attributes.

Although you set up explicit channel definitions, the channel negotiations carried out when a channel starts up may override one or other of the values defined. This is quite normal, and transparent, and has been arranged like this so that otherwise incompatible definitions can work together.

Auto-definition of receiver and server-connection channels

In WebSphere MQ for AIX, iSeries, HP-UX, Linux, Solaris, and Windows systems, and MQSeries V5.1 for OS/2 Warp, if there is no appropriate channel definition, then for a receiver or server-connection channel that has auto-definition enabled, a definition is created automatically. The definition is created using:

1. The appropriate model channel definition, SYSTEM.AUTO.RECEIVER or SYSTEM.AUTO.SVRCONN. The model channel definitions for auto-definition are the same as the system defaults, SYSTEM.DEF.RECEIVER and SYSTEM.DEF.SVRCONN, except for the description field, which is “Auto-defined by” followed by 49 blanks. The systems administrator can choose to change any part of the supplied model channel definitions.

2. Information from the partner system. The partner’s values are used for the channel name and the sequence number wrap value.

3. A channel exit program, which you can use to alter the values created by the auto-definition. See “Channel auto-definition exit program” on page 583.

The description is then checked to determine whether it has been altered by an auto-definition exit or because the model definition has been changed. If the first 44 characters are still “Auto-defined by” followed by 29 blanks, the queue manager name is added. If the final 20 characters are still all blanks the local time and date are added.

Once the definition has been created and stored the channel start proceeds as though the definition had always existed. The batch size, transmission size, and message size are negotiated with the partner.
Defining other objects
Before a message channel can be started, both ends must be defined (or enabled for auto-definition) at their respective queue managers. The transmission queue it is to serve must be defined to the queue manager at the sending end, and the communication link must be defined and available. In addition, it may be necessary for you to prepare other WebSphere MQ objects, such as remote queue definitions, queue manager alias definitions, and reply-to queue alias definitions, so as to implement the scenarios described in Chapter 2, “Making your applications communicate” on page 17.

For information about MQI channels, see the WebSphere MQ Clients book.

Multiple message channels per transmission queue
It is possible to define more than one channel per transmission queue, but only one of these channels can be active at any one time. This is recommended for the provision of alternative routes between queue managers for traffic balancing and link failure corrective action.

Starting a channel
A channel can be caused to start transmitting messages in one of four ways. It can be:

- Started by an operator (not receiver, cluster-receiver or server-connection channels).
- Triggered from the transmission queue (sender, and fully-qualified server channels only). You will need to prepare the necessary objects for triggering channels.
- Started from an application program (not receiver, cluster-receiver or server-connection channels).
- Started remotely from the network by a sender, cluster-sender, requester, server, or client-connection channel. Receiver, cluster-receiver and possibly server and requester channel transmissions, are started this way; so are server-connection channels. The channels themselves must already be started (that is, enabled).

Note: Because a channel is ‘started’ it is not necessarily transmitting messages, but, rather, it is ‘enabled’ to start transmitting when one of the four events described above occurs. The enabling and disabling of a channel is achieved using the START and STOP operator commands.

Channel states
Figure 29 on page 62 shows the hierarchy of all possible channel states, and Figure 30 on page 63 shows the links between them. These apply to all types of message channel. On WebSphere MQ for AIX, iSeries, HP-UX, Linux, Solaris, and Windows systems, and MQSeries V5.1 for OS/2 Warp, these states apply also to server-connection channels.
Channel control function

Current and active
The channel is “current” if it is in any state other than inactive. A current channel is “active” unless it is in RETRYING, STOPPED, or STARTING state.
Notes:
1. When a channel is in one of the six states highlighted in Figure 30 (INITIALIZING, BINDING, REQUESTING, RUNNING, PAUSED, or STOPPING), it is consuming resource and a process or thread is running; the
channel is active. (INITIALIZING occurs on z/OS and on WebSphere MQ for AIX, iSeries, HP-UX, Linux, Solaris, and Windows systems, and MQSeries V5.1 for Compaq Tru64 UNIX, and OS/2 Warp. (PAUSED does not occur on z/OS.)

2. When a channel is in STOPPED state, the session may be active because the next state is not yet known.

Specifying the maximum number of current channels: You can specify the maximum number of channels that can be current at one time. This is the number of channels that have entries in the channel status table, including channels that are retrying and channels that are disabled (that is, stopped). Specify this in the channel initiator parameter module for z/OS, the queue manager initialization file for iSeries, the queue manager configuration file for OS/2, Compaq NonStop Kernel, and UNIX systems, or the registry for Windows. For more information about the values you set using the initialization or the configuration file see Appendix D, “Configuration file stanzas for distributed queuing” on page 715. For more information about specifying the maximum number of channels, see the WebSphere MQ System Administration Guide for WebSphere MQ for AIX, HP-UX, Linux, Solaris, and Windows systems, and MQSeries V5.1 for Compaq Tru64 UNIX, and OS/2 Warp, the WebSphere MQ for iSeries V5.3 System Administration book for WebSphere MQ for iSeries, or the WebSphere MQ for z/OS Concepts and Planning Guide for information relating to your platform.

Notes:
1. On WebSphere MQ for AIX, iSeries, HP-UX, Linux, Solaris, Windows systems, and z/OS, and MQSeries V5.1 for Compaq Tru64 UNIX, and OS/2 Warp, server-connection channels are included in this number.
2. A channel must be current before it can become active. If a channel is started, but cannot become current, the start fails.
3. If you are using CICS for distributed queuing on z/OS, you cannot specify the maximum number of channels.

Specifying the maximum number of active channels: You can also specify the maximum number of active channels (except on WebSphere MQ for z/OS using CICS). You can do this to prevent your system being overloaded by a large number of starting channels. If you use this method, you should set the disconnect interval attribute to a low value to allow waiting channels to start as soon as other channels terminate.

Each time a channel that is retrying attempts to establish connection with its partner, it must become an active channel. If the attempt fails, it remains a current channel that is not active, until it is time for the next attempt. The number of times that a channel will retry, and how often, is determined by the retry count and retry interval channel attributes. There are short and long values for both these attributes. See Chapter 6, “Channel attributes” on page 77 for more information.

When a channel has to become an active channel (because a START command has been issued, or because it has been triggered, or because it is time for another retry attempt), but is unable to do so because the number of active channels is already at the maximum value, the channel waits until one of the active slots is freed by another channel instance ceasing to be active. If, however, a channel is starting because it is being initiated remotely, and there are no active slots available for it at that time, the remote initiation is rejected.

Whenever a channel, other than a requester channel, is attempting to become active, it goes into the STARTING state. This is true even if there is an active slot
immediately available, although in this case it will only be in STARTING state for a very short time. However, if the channel has to wait for an active slot, it is in STARTING state while it is waiting.

Requester channels do not go into STARTING state. If a requester channel cannot start because the number of active channels is already at the limit, the channel ends abnormally.

Whenever a channel, other than a requester channel, is unable to get an active slot, and so waits for one, a message is written to the log or the z/OS console, and an event is generated. When a slot is subsequently freed and the channel is able to acquire it, another message and event are generated. Neither of these events and messages are generated if the channel is able to acquire a slot straightaway.

If a STOP CHANNEL command is issued while the channel is waiting to become active, the channel goes to STOPPED state. A Channel-Stopped event is raised as usual.

On WebSphere MQ for AIX, iSeries, HP-UX, Linux, Solaris, Windows systems, and z/OS, and MQSeries V5.1 for OS/2 Warp, server-connection channels are included in the maximum number of active channels.

For more information about specifying the maximum number of active channels, see the WebSphere MQ System Administration Guide book for WebSphere MQ for AIX, iSeries, HP-UX, Solaris, and Windows systems, and MQSeries for Compaq Tru64 UNIX, and OS/2 Warp, the WebSphere MQ for iSeries V5.3 System Administration book for WebSphere MQ for iSeries, or the WebSphere MQ for z/OS Concepts and Planning Guide for the z/OS platform.

Channel errors

Errors on channels cause the channel to stop further transmissions. If the channel is a sender or server, it goes to RETRY state because it is possible that the problem may clear itself. If it cannot go to RETRY state, the channel goes to STOPPED state. For sending channels, the associated transmission queue is set to GET(DISABLED) and triggering is turned off. (A STOP command with STATUS(STOPPED) takes the side that issued it to STOPPED state; only expiry of the disconnect interval or a STOP command with STATUS(INACTIVE) will make it end normally and become inactive.) Channels that are in STOPPED state need operator intervention before they will restart (see “Restarting stopped channels” on page 69).

Note: For Compaq OpenVMS Alpha, OS/2 Warp, iSeries, UNIX systems, Compaq NonStop Kernel, and Windows systems, a channel initiator must be running for retry to be attempted. If the channel initiator is not available, the channel becomes inactive and must be manually restarted. If you are using a script to start the channel, ensure the channel initiator is running before you try to run the script. On platforms other than AIX, Compaq Tru64 UNIX, HP-UX, iSeries, OS/2 Warp, Solaris, and Windows systems, the channel initiator must be monitoring the initiation queue specified in the transmission queue that the channel is using.

“Long retry count (LONGRTY)” on page 88 describes how retrying works. If the error clears, the channel restarts automatically, and the transmission queue is re-enabled. If the retry limit is reached without the error clearing, the channel goes to STOPPED state. A stopped channel must be restarted manually by the operator. If the error is still present, it does not retry again. When it does start successfully, the transmission queue is re-enabled.
On WebSphere MQ for AIX, iSeries, HP-UX, Linux, Solaris, Windows systems, and z/OS without CICS, and MQSeries V5.1 for OS/2 Warp, if the channel initiator or queue manager stops while a channel is in RETRYING or STOPPED status, the channel status is remembered when the channel initiator or queue manager is restarted.

On OS/2 Warp, Windows systems, iSeries, Compaq NonStop Kernel, and UNIX systems, if a channel is unable to put a message to the target queue because that queue is full or put inhibited, the channel can retry the operation a number of times (specified in the message-retry count attribute) at a given time interval (specified in the message-retry interval attribute). Alternatively, you can write your own message-retry exit that determines which circumstances cause a retry, and the number of attempts made. The channel goes to PAUSED state while waiting for the message-retry interval to finish.

See Chapter 6, “Channel attributes” on page 77 for information about the channel attributes, and Chapter 45, “Channel-exit programs” on page 571 for information about the message-retry exit.

Checking that the other end of the channel is still available

In WebSphere MQ for AIX, iSeries, HP-UX, Linux, Solaris, Windows systems, and z/OS without CICS, and MQSeries V5.1 for OS/2 Warp, you can use the heartbeat-interval channel attribute to specify that flows are to be passed from the sending MCA when there are no messages on the transmission queue. This is described in “Heartbeat interval (HBINT)” on page 87.

In WebSphere MQ for z/OS, if you are using TCP as the transport protocol, you can also specify a value for the KeepAlive Interval channel attribute (KAINT). You are recommended to give the KeepAliveInterval a higher value than the heartbeat interval, and a smaller value than the disconnect value. You can use this attribute to specify a time-out value for each channel. This is described in “KeepAlive Interval (KAINT)” on page 87.

In WebSphere MQ for AIX, iSeries, HP-UX, Linux, Solaris, Windows systems, and MQSeries V5.1 for OS/2 Warp, if you are using TCP as your transport protocol, you can set keepalive=yes in the qm.ini file. If you specify this option, TCP periodically checks that the other end of the connection is still available, and if it is not, the channel is terminated.

In WebSphere MQ for AIX, iSeries, HP-UX, Linux, Solaris, and Windows systems, and MQSeries V5.1 for OS/2 Warp, if you are using TCP as your transport protocol, the receiving end of inactive connections are also closed if no data is received for a period of time. This period of time is determined according to the HBINT (heartbeat interval) value.

The time-out value is set as follows:
1. For an initial number of flows, before any negotiation has taken place, the timeout is twice the HBINT value from the channel definition.
2. When the channels have negotiated a HBINT value, if HBINT is set to less than 60 seconds, the timeout is set to twice this value. If HBINT is set to 60 seconds or more, the timeout value is set to 60 seconds greater than the value of HBINT.

Notes:
1. If either of the above values is zero, there is no timeout.
2. For connections that do not support heartbeats, the HBINT value is negotiated to zero in step 2 and hence there is no timeout, so we must use TCP/IP KEEPALIVE.

3. For client connections, heartbeats are only flowed from the server when the client issues an MQGET call with wait; none are flowed during other MQI calls. Therefore, you are not recommended to set the heartbeat interval too small for client channels. For example, if the heartbeat is set to ten seconds, an MQCMIT call will fail (with MQRC_CONNECTION_BROKEN) if it takes longer than twenty seconds to commit because no data will have been flowed during this time. This can happen with large units of work. However, it should not happen if appropriate values are chosen for the heartbeat interval because only MQGET with wait should take significant periods of time.

4. Aborting the connection after twice the heartbeat interval is valid because a data or heartbeat flow is expected at least every heartbeat interval. Setting the heartbeat interval too small, however, can cause problems, especially if you are using channel exits. For example, if the HBINT value is one second, and a send or receive exit is used, the receiving end will only wait for two seconds before aborting the channel. If the MCA is performing a task such as encrypting the message, this value might be too short.

If you have unreliable channels that are suffering from TCP errors, use of SO_KEEPALIVE will mean that your channels are more likely to recover.

You can specify time intervals to control the behavior of the SO_KEEPALIVE option. When you change the time interval, only TCP/IP channels started after the change are affected. The value that you choose for the time interval should be less than the value of the disconnect interval for the channel.

For more information about using the SO_KEEPALIVE option on z/OS, see WebSphere MQ for z/OS Concepts and Planning Guide. For other platforms, see the chapter about setting up communications for your platform in this manual.

Adopting an MCA

If a channel suffers a communications failure, the receiver channel could be left in a ‘communications receive’ state. When communications are re-established the sender channel attempts to reconnect. If the remote queue manager finds that the receiver channel is already running it does not allow another version of the same receiver channel to be started. This problem requires user intervention to rectify the problem or the use of system keepalive.

The Adopt MCA function solves the problem automatically. It enables WebSphere MQ to cancel a receiver channel and to start a new one in its place.

The function can be set up with various options. For more information see WebSphere MQ for z/OS System Setup Guide or the appropriate equivalent publication for your platforms.

Stopping and quiescing channels

Message channels are designed to be long-running connections between queue managers with orderly termination controlled only by the disconnect interval channel attribute. This mechanism works well unless the operator needs to terminate the channel before the disconnect time interval expires. This can occur in the following situations:

- System quiesce
- Resource conservation
Unilateral action at one end of a channel

In this case, an operator command is provided to allow you to stop the channel. The command provided varies by platform, as follows:

For z/OS without CICS:
The STOP CHANNEL MQSC command or the Stop a channel panel

For z/OS using CICS:
The Stop option on the Message Channel List panel

For OS/2, Windows systems, Compaq OpenVMS Alpha, Compaq NonStop Kernel, and UNIX systems:
The STOP CHANNEL MQSC or PCF command

For iSeries:
ENDMQMCHL or the END option on the WRKMQMCHL panel

For VSE/ESA:
The CLOSE command from the MQMMSC panel or MQCL transaction closes (rather than stops) the channel.

There are three options for stopping channels using these commands:

QUIESCE
The QUIESCE option attempts to end the current batch of messages before stopping the channel.

FORCE
The FORCE option attempts to stop the channel immediately and may require the channel to resynchronize when it restarts because the channel may be left in doubt.

TERMINATE
The TERMINATE option attempts to stop the channel immediately, and terminates the channel’s thread or process.

Note that all of these options leave the channel in a STOPPED state, requiring operator intervention to restart it.

Stopping the channel at the sending end is quite effective but does require operator intervention to restart. At the receiving end of the channel, things are much more difficult because the MCA is waiting for data from the sending side, and there is no way to initiate an orderly termination of the channel from the receiving side; the stop command is pending until the MCA returns from its wait for data.

Consequently there are three recommended ways of using channels, depending upon the operational characteristics required:

• If you want your channels to be long running, you should note that there can be orderly termination only from the sending end. When channels are interrupted, that is, stopped, operator intervention (a START CHANNEL command) is required in order to restart them.

• If you want your channels to be active only when there are messages for them to transmit, you should set the disconnect interval to a fairly low value. Note that the default setting is quite high and so is not recommended for channels where this level of control is required. Because it is difficult to interrupt the receiving channel, the most economical option is to have the channel
automatically disconnect and reconnect as the workload demands. For most
channels, the appropriate setting of the disconnect interval can be established
heuristically.

- For WebSphere MQ for AIX, iSeries, HP-UX, Linux, Solaris, Windows systems,
  and z/OS without CICS, and MQSeries V5.1 for OS/2 Warp, you can use the
  heartbeat-interval attribute to cause the sending MCA to send a heartbeat flow
to the receiving MCA during periods in which it has no messages to send. This
releases the receiving MCA from its wait state and gives it an opportunity to
quiesce the channel without waiting for the disconnect interval to expire. Give
the heartbeat interval a lower value than the value of the disconnect interval.

Notes:
1. You are advised to set the disconnect interval to a low value, or to use
   heartbeats, for server channels. This is to allow for the case where the
   requester channel ends abnormally (for example, because the channel was
canceled) when there are no messages for the server channel to send. In this
case, the server does not detect that the requester has ended (it will only do
this the next time it tries to send a message to the requester). While the
server is still running, it holds the transmission queue open for exclusive
input in order to get any more messages that arrive on the queue. If an
attempt is made to restart the channel from the requester, the start request
receives an error because the server still has the transmission queue open for
exclusive input. It is necessary to stop the server channel, and then restart
the channel from the requester again.

2. On WebSphere MQ for AIX, iSeries, HP-UX, Linux, Solaris, Windows
   systems, and z/OS without CICS, and MQSeries V5.1 for Compaq Tru64
   UNIX, and OS/2 Warp, server-connection channels can also be stopped like
   receiver channels.

Restarting stopped channels

When a channel goes into STOPPED state (either because you have stopped the
channel manually using one of the methods given in “Stopping and quiescing
channels” on page 67, or because of a channel error) you have to restart the
channel manually.

To do this, issue one of the following commands:

For WebSphere MQ for z/OS without CICS:
The START CHANNEL MQSC command or the Start a channel panel

For WebSphere MQ for z/OS using CICS:
The Start option on the Message Channel List panel

For WebSphere MQ for UNIX systems and Windows systems, and MQSeries for
OS/2 Warp, Compaq OpenVMS Alpha, Compaq NonStop Kernel, :
The START CHANNEL MQSC or PCF command

For WebSphere MQ for iSeries:
The START command on the WRKMQMCHL panel, the STRMQMCHL
command, or the START CHANNEL MQSC or PCF command

For MQSeries for VSE/ESA:
The OPEN command from the MQMMSC panel or MQCL transaction
opens (rather than restarts) the channel.

For sender or server channels, when the channel entered the STOPPED state, the
associated transmission queue was set to GET(DISABLED) and triggering was set
off. When the start request is received, these attributes are reset automatically. On
WebSphere MQ for AIX, iSeries, HP-UX, Linux, Solaris, Windows systems, and z/OS without CICS, and MQSeries V5.1 for OS/2 Warp, if the channel initiator or queue manager stops while a channel is in RETRYING or STOPPED status, the channel status is remembered when the channel initiator or queue manager is restarted. On other platforms, if the channel initiator or queue manager is restarted the status is lost and you have to alter the queue attributes manually to re-enable triggering of the channel.

**Note:** If you are using CICS for distributed queuing on z/OS, these queue attributes are not reset automatically; you always have to alter them manually when you restart a channel.

### In-doubt channels

An in-doubt channel is a channel that is in doubt with a remote channel about which messages have been sent and received. Note the distinction between this and a queue manager being in doubt about which messages should be committed to a queue.

You can reduce the opportunity for a channel to be placed in doubt by using the Batch Heartbeat channel parameter (BATCHHB). When a value for this parameter is specified, a sender channel checks that the remote channel is still active before taking any further action. If no response is received the receiver channel is considered to be no longer active. The messages can be rolled-back, and re-routed, and the sender-channel is not put in doubt. This reduces the time when the channel could be placed in doubt to the period between the sender channel verifying that the receiver channel is still active, and verifying that the receiver channel has received the sent messages. See Chapter 6, “Channel attributes” on page 77 for more information on the batch heartbeat parameter.

In-doubt channel problems are usually resolved automatically. Even when communication is lost, and a channel is placed in doubt with a message batch at the sender whose receipt status is unknown, the situation is resolved when communication is re-established. Sequence number and LUWID records are kept for this purpose. The channel is in doubt until LUWID information has been exchanged, and only one batch of messages can be in doubt for the channel.

You can, when necessary, resynchronize the channel manually. The term manual includes use of operators or programs that contain WebSphere MQ system management commands. The manual resynchronization process works as follows. This description uses MQSC commands, but you can also use the PCF equivalents.

1. Use the DISPLAY CHSTATUS command to find the last-committed logical unit of work ID (LUWID) for each side of the channel. Do this using the following commands:
   - For the in-doubt side of the channel:
     ```
     DISPLAY CHSTATUS(name) SAVED CURLUWID
     ```
   - You can use the CONNAME and XMITQ parameters to further identify the channel.
   - For the receiving side of the channel:
     ```
     DISPLAY CHSTATUS(name) SAVED LSTLUWID
     ```
   - You can use the CONNAME parameter to further identify the channel.
The commands are different because only the sending side of the channel can be in doubt. The receiving side is never in doubt.

On WebSphere MQ for iSeries, the DISPLAY CHSTATUS command can be executed from a file using the STRMQMMQSC command or the Work with MQM Channel Status CL command, WRKMQMCHST.

2. If the two LUWIDs are the same, the receiving side has committed the unit of work that the sender considers to be in doubt. The sending side can now remove the in-doubt messages from the transmission queue and re-enable it. This is done with the following channel RESOLVE command:

   RESOLVE CHANNEL(name) ACTION(COMMIT)

3. If the two LUWIDs are different, the receiving side has not committed the unit of work that the sender considers to be in doubt. The sending side needs to retain the in-doubt messages on the transmission queue and re-send them. This is done with the following channel RESOLVE command:

   RESOLVE CHANNEL(name) ACTION(BACKOUT)

   On WebSphere MQ for iSeries, you can use the Resolve MQM Channel command, RSVMQMCHL.

   Once this process is complete the channel is no longer in doubt. The transmission queue can now be used by another channel, if required.

Problem determination

There are two distinct aspects to problem determination:

- Problems discovered when a command is being submitted
- Problems discovered during operation of the channels

Command validation

Commands and panel data must be free from errors before they are accepted for processing. Any errors found by the validation are immediately notified to the user by error messages.

Problem diagnosis begins with the interpretation of these error messages and taking the recommended corrective action.

Processing problems

Problems found during normal operation of the channels are notified to the system console or the system log. Problem diagnosis begins with the collection of all relevant information from the log, and continues with analysis to identify the problem.

Confirmation and error messages are returned to the terminal that initiated the commands, when possible.

Messages and codes

Where provided, the Messages and Codes manual of the particular platform can help with the primary diagnosis of the problem.

What happens when a message cannot be delivered?

Figure 31 on page 72 shows the processing that occurs when an MCA is unable to put a message to the destination queue. (Note that the options shown do not apply on all platforms.)
As shown in the figure, the MCA can do several things with a message that it cannot deliver. The action taken is determined by options specified when the channel is defined and on the MQPUT report options for the message.

1. **Message-retry**
   If the MCA is unable to put a message to the target queue for a reason that could be transitory (for example, because the queue is full), the MCA has the option to wait and retry the operation later. You can determine if the MCA waits, for how long, and how many times it retries.
   - You can specify a message-retry time and interval for MQPUT errors when you define your channel. If the message cannot be put to the destination queue because the queue is full, or is inhibited for puts, the MCA retries the operation the number of times specified, at the time interval specified.
   - You can write your own message-retry exit. The exit enables you to specify under what conditions you want the MCA to retry the MQPUT or MQOPEN operation. Specify the name of the exit when you define the channel.

   Message-retry is not available on WebSphere MQ for z/OS, or MQSeries for VSE/ESA.

2. **Return-to-sender**
   If message-retry was unsuccessful, or a different type of error was encountered, the MCA can send the message back to the originator.

   To enable this, you need to specify the following options in the message descriptor when you put the message to the original queue:
   - The MQRO_EXCEPTION_WITH_FULL_DATA report option
Undelivered messages

- The MQRO_DISCARD_MSG report option
- The name of the reply-to queue and reply-to queue manager

If the MCA is unable to put the message to the destination queue, it generates an exception report containing the original message, and puts it on a transmission queue to be sent to the reply-to queue specified in the original message. (If the reply-to queue is on the same queue manager as the MCA, the message is put directly to that queue, not to a transmission queue.)

Return-to-sender is not available on MQSeries for VSE/ESA.

3. Dead-letter queue

If a message cannot be delivered or returned, it is put on to the dead-letter queue (DLQ). You can use the DLQ handler to process the message. This is described in the [WebSphere MQ System Administration Guide](#) book for WebSphere MQ for AIX, HP-UX, Linux, Solaris, and Windows systems, and MQSeries V5.1 for OS/2 Warp, and in the [WebSphere MQ for z/OS System Administration Guide](#) for z/OS.

If the dead-letter queue is not available, the sending MCA leaves the message on the transmission queue, and the channel stops. On a fast channel, nonpersistent messages that cannot be written to a dead-letter queue are lost.

On WebSphere MQ Version 5 products, if no local dead-letter queue is defined, the remote queue is not available or defined, and there is no remote dead-letter queue, the channel stops abnormally, and messages are not rolled back to the sending transmission queue. You must resolve the channel using the COMMIT or BACKOUT functions.

Initialization and configuration files

The handling of channel initialization data depends on your WebSphere MQ platform.

z/OS without CICS

In WebSphere MQ for z/OS without CICS, initialization and configuration information is in the channel initiator parameter module CSQXPARM. You can also put commands in the CSQINPX initialization input data set, which is processed every time you start the channel initiator if you specify the optional DD statement CSQINPX in the channel initiator started task procedure. See [WebSphere MQ for z/OS Concepts and Planning Guide](#) for information about both of these.

z/OS using CICS

In WebSphere MQ for z/OS using CICS there is no channel initiator.

Windows systems

On WebSphere® MQ for Windows systems, the registry file holds basic configuration information about the WebSphere MQ installation. That is, information relevant to all of the queue managers on the WebSphere MQ system and also information relating to individual queue managers.
Initialization and configuration files

OS/2, Compaq OpenVMS Alpha, Compaq NonStop Kernel, iSeries, and UNIX systems

On MQSeries for OS/2 Warp, MQSeries for Compaq OpenVMS Alpha, Compaq NonStop Kernel, and WebSphere MQ for iSeries and WebSphere MQ on UNIX systems, there are configuration files to hold basic configuration information about the WebSphere MQ installation.

There are two configuration files: one applies to the machine, the other applies to an individual queue manager.

WebSphere MQ configuration file
This holds information relevant to all of the queue managers on the WebSphere MQ system. The file is called MQSINI on Compaq NonStop Kernel and mqs.ini on other platforms. It is fully described in the WebSphere MQ System Administration Guide book for WebSphere MQ for AIX, HP-UX, and Solaris, and MQSeries for OS/2 Warp, and in the WebSphere MQ for iSeries V5.3 System Administration book for iSeries.

Queue manager configuration file
The queue manager configuration file holds configuration information relating to one particular queue manager. The file is called QMINI on Compaq NonStop Kernel, and qm.ini on other platforms.

It is created during queue manager creation and may hold configuration information relevant to any aspect of the queue manager. Information held in the file includes details of how the configuration of the log differs from the default in WebSphere MQ configuration file.

The queue manager configuration file is held in the root of the directory tree occupied by the queue manager. On WebSphere MQ for Windows, the information is held in the registry. For example, for the DefaultPath attributes, the queue manager configuration files for a queue manager called QMNAME would be:

For OS/2:
c:\mqm\qmgrs\QMNAME\qm.ini

For UNIX systems:
/var/mqm/qmgrs/QMNAME/qm.ini

For Compaq OpenVMS Alpha:
mq_root:[mqm.qmgrs.QMNAME]qm.ini

For Compaq NonStop Kernel:
The file is held in the subvolume of the queue manager. For example, the path and name for a configuration file for a queue manager called QMNAME could be $VOLUME.QMNAMED.QMINI.

An excerpt of a qm.ini file follows. It specifies that the TCP/IP listener is to listen on port 2500, the maximum number of current channels is to be 200 and the maximum number of active channels is to be 100.

TCP:
  Port=2500
CHANNELS:
  MaxChannels=200
  MaxActiveChannels=100
In MQSeries V5.2 and WebSphere MQ, you can specify a range of TCP/IP ports to be used by an outbound channel. One method is to use the qm.ini file to specify the start and end of a range of port values. The example below shows a qm.ini file specifying a range of channels:

```
TCP:
    StrPort=2500
    EndPort=3000

CHANNELS:
    MaxChannels=200
    MaxActiveChannels=100
```

If you specify a value for StrPort or EndPort then you must specify a value for both. The value of EndPort must always be greater than the value of StrPort.

The channel tries to use each of the port values in the range specified. When the connection is successful, the port value is the port that the channel then uses.

**Note:** For Compaq NonStop Kernel, the format of the qm.ini file is slightly different. For more details about this, see the MQSeries for Compaq NonStop Kernel System Administration.

For iSeries:
/QIBM/UserData/mqm/qmgrs/QMNAME/qm.ini

For more information about qm.ini files see Appendix D, “Configuration file stanzas for distributed queuing” on page 715.

### VSE/ESA

There is no qm.ini file on VSE/ESA. Instead, use the Configuration main menu on the MQMCCFG panel to configure the queue manager.

### Data conversion

A WebSphere MQ message consists of two parts:

- Control information in a message descriptor
- Application data

Either of the two parts may require data conversion when sent between queues on different queue managers. For information about data conversion, see the WebSphere MQ Application Programming Guide.

### Writing your own message channel agents

WebSphere MQ products allow you to write your own message channel agent (MCA) programs or to install one from an independent software vendor. You might want to do this to make a WebSphere MQ product interoperate over your own, proprietary communications protocol or to send messages over a protocol that WebSphere MQ does not support. (You cannot write your own MCA to interoperate with a WebSphere MQ-supplied MCA at the other end.)

If you decide to use an MCA that was not supplied by WebSphere MQ, you need to consider the following.

**Message sending and receiving**

You need to write a sending application that gets messages from wherever your application puts them, for example from a transmission queue.
Writing message channel agents

the *WebSphere MQ Application Programming Reference* book), and sends them out on a protocol with which you want to communicate. You also need to write a receiving application that takes messages from this protocol and puts them onto destination queues. The sending and receiving applications use the message queue interface (MQI) calls, not any special interfaces.

You need to ensure that messages are delivered once and once only. Syncpoint coordination can be used to help with this.

Channel control function
You need to provide your own administration functions to control channels. You cannot use WebSphere MQ channel administration functions either for configuring (for example, the DEFINE CHANNEL command) or monitoring (for example, DISPLAY CHSTATUS) your channels.

Initialization file
You need to provide your own initialization file, if you require one.

Application data conversion
You will probably want to allow for data conversion for messages you send to a different system. If so, use the MQGMO_CONVERT option on the MQGET call when retrieving messages from wherever your application puts them, for example the transmission queue.

User exits
Consider whether you need user exits. If so, you can use the same interface definitions that WebSphere MQ uses.

Triggering
If your application puts messages to a transmission queue, you can set up the transmission queue attributes so that your sending MCA is triggered when messages arrive on the queue.

Channel initiator
You may need to provide your own channel initiator.
Chapter 6. Channel attributes

The previous chapters have introduced the basic concepts of the product, the business perspective basis of its design, its implementation, and the control features.

This chapter describes the channel attributes held in the channel definitions. This is product-sensitive programming interface information.

You choose the attributes of a channel to be optimal for a given set of circumstances for each channel. However, when the channel is running, the actual values may have changed during startup negotiations. See “Preparing channels” on page 60.

Many attributes have default values, and you can use these for most channels. However, in those circumstances where the defaults are not optimal, refer to this chapter for guidance in selecting the correct values.

Note: In WebSphere MQ for iSeries, most parameters can be specified as *SYSDFTCCHL, which means that the value is taken from the system default channel in your system.

Channel attributes and channel types

The channel types for WebSphere MQ channel attributes are listed in Table 7.

<table>
<thead>
<tr>
<th>Attribute field</th>
<th>SDR</th>
<th>SVR</th>
<th>RCVR</th>
<th>RQSTR</th>
<th>CLNT-CONN</th>
<th>SVR-CONN</th>
<th>CLUS-SDR</th>
<th>CLUS-RCVR</th>
</tr>
</thead>
<tbody>
<tr>
<td>Batch heartbeat interval</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Batch interval</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Batch size</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>Channel name</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>Cluster</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cluster namelist</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Channel type</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Connection name</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Convert message</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Description</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Disconnect interval</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Heartbeat interval</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Keepalive interval</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Local address</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Long retry count</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Long retry interval</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LU 6.2 Transaction program name</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Maximum message length</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

© Copyright IBM Corp. 1993, 2002
## Channel attributes

### Table 7. Channel attributes for the channel types  (continued)

<table>
<thead>
<tr>
<th>Attribute field</th>
<th>SDR</th>
<th>SVR</th>
<th>RCVR</th>
<th>RQSTR</th>
<th>CLNT-CONN</th>
<th>SVR-CONN</th>
<th>CLUS-SDR</th>
<th>CLUS-RCVR</th>
</tr>
</thead>
<tbody>
<tr>
<td>Message channel agent type</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Message channel agent user</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Message exit name</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Message exit user data</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Message-retry exit name</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Message-retry exit user data</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Message retry count</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Message retry interval</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mode name</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Network-connection priority</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Nonpersistent message speed</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Password</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PUT authority</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Queue manager name</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Receive exit</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Receive exit user data</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Security exit</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Security exit user data</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Send exit</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Send exit user data</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sequence number wrap</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Short retry interval</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Short retry count</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SSL Cipher Specification</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SSL Client Authentication</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SSL Peer</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Transport type</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Transmission queue</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>User ID</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Channel attributes in alphabetical order

WebSphere MQ for some platforms may not implement all the attributes shown in the list. Exceptions and platform differences are mentioned in the individual attribute descriptions, where relevant.

The keyword that you can specify in MQSC is shown in brackets for each attribute. (Attributes that apply only to WebSphere MQ for z/OS with CICS do not have MQSC keywords.)
The attributes are arranged in alphabetical order, as follows:

<table>
<thead>
<tr>
<th>Attribute</th>
<th>See page...</th>
</tr>
</thead>
<tbody>
<tr>
<td>Auto start (AUTOSTART)</td>
<td>80</td>
</tr>
<tr>
<td>Alter date (ALTDATE)</td>
<td>80</td>
</tr>
<tr>
<td>Alter time (ALTTIME)</td>
<td>80</td>
</tr>
<tr>
<td>Batch Heartbeat Interval (BATCHHB)</td>
<td>80</td>
</tr>
<tr>
<td>Batch interval (BATCHINT)</td>
<td>81</td>
</tr>
<tr>
<td>Batch size (BATCHSZ)</td>
<td>81</td>
</tr>
<tr>
<td>Channel name (CHANNEL)</td>
<td>82</td>
</tr>
<tr>
<td>Channel type (CHLTYPE)</td>
<td>83</td>
</tr>
<tr>
<td>CICS profile name</td>
<td>83</td>
</tr>
<tr>
<td>Cluster (CLUSTER)</td>
<td>83</td>
</tr>
<tr>
<td>Cluster namelist (CLUSNL)</td>
<td>83</td>
</tr>
<tr>
<td>Connection name (CONNAME)</td>
<td>84</td>
</tr>
<tr>
<td>Convert message (CONVERT)</td>
<td>85</td>
</tr>
<tr>
<td>Description (DESCR)</td>
<td>86</td>
</tr>
<tr>
<td>Disconnect interval (DISCINT)</td>
<td>87</td>
</tr>
<tr>
<td>KeepAlive Interval (KAINT)</td>
<td>92</td>
</tr>
<tr>
<td>Local Address (LOCLADDR)</td>
<td>88</td>
</tr>
<tr>
<td>Long retry count (LONGLRTY)</td>
<td>88</td>
</tr>
<tr>
<td>Long retry count (LONGLRTY)</td>
<td>88</td>
</tr>
<tr>
<td>Long retry interval (LONGTMR)</td>
<td>89</td>
</tr>
<tr>
<td>LU 6.2 mode name (MODENAME)</td>
<td>89</td>
</tr>
<tr>
<td>LU 6.2 transaction program name (TPNAME)</td>
<td>89</td>
</tr>
<tr>
<td>Maximum message length (MAXMSGL)</td>
<td>90</td>
</tr>
<tr>
<td>Maximum transmission size</td>
<td>90</td>
</tr>
<tr>
<td>Message channel agent name (MCANAME)</td>
<td>91</td>
</tr>
<tr>
<td>Message channel agent type (MCATYPE)</td>
<td>91</td>
</tr>
<tr>
<td>Message channel agent user identifier (MCAUSER)</td>
<td>91</td>
</tr>
<tr>
<td>Message exit name (MSGEXIT)</td>
<td>92</td>
</tr>
<tr>
<td>Message exit user data (MSGDATA)</td>
<td>92</td>
</tr>
<tr>
<td>Message-retry exit name (MREXIT)</td>
<td>92</td>
</tr>
<tr>
<td>Message-retry exit user data (MRDATA)</td>
<td>92</td>
</tr>
<tr>
<td>Message retry count (MRRTY)</td>
<td>92</td>
</tr>
<tr>
<td>Message retry interval (MRTMR)</td>
<td>93</td>
</tr>
<tr>
<td>Nonpersistent message speed (NPMSPEED)</td>
<td>93</td>
</tr>
<tr>
<td>Network-connection priority (NETPRTY)</td>
<td>93</td>
</tr>
<tr>
<td>Password (PASSWORD)</td>
<td>93</td>
</tr>
<tr>
<td>PUT authority (PUTAUT)</td>
<td>94</td>
</tr>
<tr>
<td>Queue manager name (QMNAME)</td>
<td>95</td>
</tr>
<tr>
<td>Receive exit name (RCVEXIT)</td>
<td>95</td>
</tr>
<tr>
<td>Receive exit user data (RCVDATA)</td>
<td>95</td>
</tr>
<tr>
<td>Security exit name (SCYEXIT)</td>
<td>96</td>
</tr>
<tr>
<td>Security exit user data (SCYDATA)</td>
<td>96</td>
</tr>
<tr>
<td>Send exit name (SENDEXIT)</td>
<td>96</td>
</tr>
<tr>
<td>Send exit user data (SENDDATA)</td>
<td>96</td>
</tr>
<tr>
<td>Sequence number wrap (SEQWRAP)</td>
<td>96</td>
</tr>
<tr>
<td>Sequential delivery</td>
<td>97</td>
</tr>
<tr>
<td>Short retry count (SHORTRTY)</td>
<td>97</td>
</tr>
<tr>
<td>Short retry interval (SHORTTMR)</td>
<td>97</td>
</tr>
<tr>
<td>SSL Cipher Specification (SSLCIPH)</td>
<td>98</td>
</tr>
<tr>
<td>SSL Client Authentication (SSLCAUTH)</td>
<td>98</td>
</tr>
<tr>
<td>SSL Peer (SSLPEER)</td>
<td>98</td>
</tr>
<tr>
<td>Target system identifier</td>
<td>99</td>
</tr>
</tbody>
</table>
Channel attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>See page...</th>
</tr>
</thead>
<tbody>
<tr>
<td>Transmission queue name (XMITQ)</td>
<td>99</td>
</tr>
<tr>
<td>Transport type (TRPTYPE)</td>
<td>99</td>
</tr>
<tr>
<td>User ID (USERID)</td>
<td>100</td>
</tr>
</tbody>
</table>

**Alter date (ALTDATE)**

This is the date on which the definition was last altered, in the form yyyy-mm-dd.

This parameter is supported on AIX, HP-UX, Linux, OS/2 Warp, z/OS, iSeries, Solaris, and Windows systems only.

**Alter time (ALTTIME)**

This is the time at which the definition was last altered, in the form hh:mm:ss.

This parameter is supported on AIX, HP-UX, Linux, OS/2 Warp, z/OS, iSeries, Solaris, and Windows systems only.

**Auto start (AUTOSTART)**

In MQSeries for Compaq NonStop Kernel there is no SNA listener process. Each channel initiated from a remote system must have its own, unique TP name on which it can listen. Such channels must be defined to MQSC with the attribute AUTOSTART(ENABLED) to ensure that there is an LU 6.2 responder process listening on this TP name whenever the queue manager is started.

SNA channels defined AUTOSTART(DISABLED) do not listen for incoming SNA requests. LU 6.2 responder processes are not started for such channels.

**Batch Heartbeat Interval (BATCHHB)**

The batch heartbeat interval allows a sending channel to verify that the receiving channel is still active just before committing a batch of messages, so that if the receiving channel is not active, the batch can be backed out rather than becoming in-doubt, as would otherwise be the case. By backing out the batch, the messages remain available for processing so they could, for example, be redirected to another channel.

If the sending channel has had a communication from the receiving channel within the batch heartbeat interval, the receiving channel is assumed to be still active, otherwise a ‘heartbeat’ is sent to the receiving channel to check.

The value must be in the range zero through 999 999. A value of zero indicates that batch heartbeating is not used.

This parameter is valid for the following channel types:

- Sender
- Server
- Cluster sender
- Cluster receiver
Batch interval (BATCHINT)

WebSphere MQ for AIX, iSeries, HP-UX, Linux, Solaris, and Windows systems, and MQSeries V5.1 for OS/2 Warp, and WebSphere MQ for z/OS without CICS, you can specify a period of time, in milliseconds, during which the channel will keep a batch open even if there are no messages on the transmission queue. You can specify any number of milliseconds, from zero through 999 999 999. The default value is zero.

If you do not specify a batch interval, the batch closes when the number of messages specified in BATCHSZ has been sent or when the transmission queue becomes empty. On lightly loaded channels, where the transmission queue frequently becomes empty the effective batch size may be much smaller than BATCHSZ.

You can use the BATCHINT attribute to make your channels more efficient by reducing the number of short batches. Be aware, however, that you may slow down the response time, because batches will last longer and messages will remain uncommitted for longer.

If you specify a BATCHINT, batches close only when one of the following conditions is met:

- The number of messages specified in BATCHSZ have been sent.
- There are no more messages on the transmission queue and a time interval of BATCHINT has elapsed while waiting for messages (since the first message of the batch was retrieved).

Note: BATCHINT specifies the total amount of time that is spent waiting for messages. It does not include the time spent retrieving messages that are already available on the transmission queue, or the time spent transferring messages.

This attribute applies only to sender, cluster-sender, server, and cluster-receiver channels.

Batch size (BATCHSZ)

The batch size is the maximum number of messages to be sent before a syncpoint is taken. The batch size does not affect the way the channel transfers messages; messages are always transferred individually, but are committed or backed out as a batch.

To improve performance, you can set a batch size to define the maximum number of messages to be transferred between two syncpoints. The batch size to be used is negotiated when a channel starts up, and the lower of the two channel definitions is taken. On some implementations, the batch size is calculated from the lowest of the two channel definitions and the two queue manager MAXUMSGS values. The actual size of a batch can be less than this; for example, a batch completes when there are no messages left on the transmission queue or the batch interval expires.

A large value for the batch size increases throughput, but recovery times are increased because there are more messages to back out and re-send. The default BATCHSZ is 50, and you are advised to try that value first. You might choose a lower value for BATCHSZ if your communications are unreliable, making the need to recover more likely.
Syncpoint procedure needs a unique logical unit of work identifier to be exchanged across the link every time a syncpoint is taken, to coordinate batch commit procedures.

If the synchronized batch commit procedure is interrupted, an in-doubt situation may arise. In-doubt situations are resolved automatically when a message channel starts up. If this resolution is not successful, manual intervention may be necessary, making use of the RESOLVE command.

Some considerations when choosing the number for batch size:

- If the number is too large, the amount of queue space taken up on both ends of the link becomes excessive. Messages take up queue space when they are not committed, and cannot be removed from queues until they are committed.
- If there is likely to be a steady flow of messages, you can improve the performance of a channel by increasing the batch size. However, this has the negative effect of increasing restart times, and very large batches may also affect performance.
- If message flow characteristics indicate that messages arrive intermittently, a batch size of 1 with a relatively large disconnect time interval may provide a better performance.
- The number may be in the range 1 through 9999. However, for data integrity reasons, channels connecting to any of the current platforms, as described in this book, should specify a batch size greater than 1. (A value of 1 is for use with Version 1 products, apart from MQSeries for MVS/ESA.)
- For z/OS using CICS it must also be at least 3 less than the value set by the ALTER QMGR MAXUMSGS command.
- Even though nonpersistent messages on a fast channel do not wait for a syncpoint, they do contribute to the batch-size count.

Channel name (CHANNEL)

Specifies the name of the channel definition. The name can contain up to 20 characters, although as both ends of a message channel must have the same name, and other implementations may have restrictions on the size, the actual number of characters may have to be smaller.

Where possible, channel names should be unique to one channel between any two queue managers in a network of interconnected queue managers.

The name must contain characters from the following list:

- Alphabetic (A-Z, a-z; note that uppercase and lowercase are significant)
- Numerics (0-9)
- Period (.)
- Forward slash (/)
- Underscore (_)
- Percentage sign (%)  

Notes:
1. Embedded blanks are not allowed, and leading blanks are ignored.
2. On systems using EBCDIC Katakana, you cannot use lowercase characters.
Channel type (CHLTYPE)

Specifies the type of the channel being defined. The possible channel types are:

**Message channel types:**
- Sender
- Server (not MQSeries for VSE/ESA)
- Cluster-sender (WebSphere MQ for z/OS without CICS, WebSphere MQ for AIX, iSeries, HP-UX, Linux, Solaris, and Windows systems, and MQSeries V5.1 for Compaq Tru64 UNIX, and OS/2 Warp only)
- Receiver
- Requester (not MQSeries for VSE/ESA)
- Cluster-receiver (WebSphere MQ for z/OS without CICS, WebSphere MQ for AIX, iSeries, HP-UX, Linux, Solaris, and Windows systems, and MQSeries V5.1 for Compaq Tru64 UNIX, and OS/2 Warp, only)

**MQI channel types:**
- Client-connection (MQSeries for OS/2 Warp, VSE/ESA, DOS, Windows 3.1, Windows 95, and Windows 98 only, and WebSphere MQ for Windows systems, and UNIX systems only)

*Note:* Client-connection channels can also be defined on z/OS for use on other platforms.
- Server-connection (not WebSphere MQ for z/OS using CICS)

The two ends of a channel must have the same name and have compatible types:
- Sender with receiver
- Requester with server
- Requester with sender (for Call_back)
- Server with receiver (server is used as a sender)
- Client-connection with server-connection
- Cluster-sender with cluster-receiver

CICS profile name

This is for z/OS using CICS only, to give extra definition for the session characteristics of the connection when CICS performs a communication session allocation, for example to select a particular COS.

The name must be known to CICS and be one to eight alphanumeric characters long.

Cluster (CLUSTER)

The name of the cluster to which the channel belongs. The maximum length is 48 characters conforming to the rules for naming WebSphere MQ objects.

This parameter is valid only for cluster-sender and cluster-receiver channels. Up to one of the resultant values of CLUSTER or CLUSNL can be nonblank. If one of the values is nonblank, the other must be blank.

This parameter is supported on AIX, HP-UX, Linux, OS/2 Warp, z/OS without CICS, iSeries, Solaris, and Windows systems only.

Cluster namelist (CLUSNL)

The name of the namelist that specifies a list of clusters to which the channel belongs.
Cluster namelist (CLUSNL)

This parameter is valid only for cluster-sender and cluster-receiver channels. Up to one of the resultant values of CLUSTER or CLUSNL can be nonblank. If one of the values is nonblank, the other must be blank.

This parameter is supported on AIX, HP-UX, Linux, OS/2 Warp, z/OS without CICS, iSeries, Solaris, and Windows systems only.

Connection name (CONNAME)

This is the communications connection identifier. It specifies the particular communications link to be used by this channel.

This attribute is required for sender channels, cluster-sender channels, cluster-receiver channels, requester channels, and client-connection channels. It does not apply to receiver or server-connection channel types.

It is optional for server channels, except on z/OS using CICS where it is required in the channel definition, but is ignored unless the server is initiating the conversation.

For z/OS using CICS this attribute names the CICS communication connection identifier for the session to be used for this channel. The name is one to four alphanumeric characters long.

Otherwise, the name is up to 48 characters for z/OS, 264 characters for other platforms, and:

If the transport type is TCP
This is either the hostname or the network address of the remote machine (or the local machine for cluster-receiver channels). For example, (MACH1.ABC.COM) or (19.22.11.162). It may include the port number, for example (MACHINE(123)). It can include the IP_name of a z/OS dynamic DNS group or a network dispatcher input port.

If the transport type is UDP
For WebSphere MQ for AIX only, UDP is an alternative to TCP. As with TCP/IP, it is either the hostname or the network address of the remote machine.

If the transport type is LU 6.2
For MQSeries for OS/2, and WebSphere MQ for iSeries, Windows systems, and UNIX systems, give the fully-qualified name of the partner LU if the TPNAME and MODENAME are specified. For other versions or if the TPNAME and MODENAME are blank, give the CPI-C side information object name as described in the section in this book about setting up communication for your platform.

On z/OS there are two forms in which to specify the value:

- Logical unit name
  The logical unit information for the queue manager, comprising the logical unit name, TP name, and optional mode name. This can be specified in one of 3 forms:
  luname, for example IGY12355
  luname/TPname, for example IGY12345/APING
  luname/TPname/modename, for example IGY12345/APINGD/#INTER
For the first form, the TP name and mode name must be specified for the TPNAME and MODENAME attributes; otherwise these attributes must be blank.

**Note:** For client-connection channels, only the first form is allowed.

- **Symbolic name**
  The symbolic destination name for the logical unit information for the queue manager, as defined in the side information data set. The TPNAME and MODENAME attributes must be blank.

  **Note:** For cluster-receiver channels, the side information is on the other queue managers in the cluster. Alternatively, in this case it can be a name that a channel auto-definition exit can resolve into the appropriate logical unit information for the local queue manager.

The specified or implied LU name can be that of a VTAM® generic resources group.

For Compaq OpenVMS Alpha, specify the Gateway Node name, the Access Name to the channel program, and the TPNAME used to invoke the remote program. For example:

```
CONNAME('SNAGWY.VMSREQUESTER(HOSTVR)')
```

For Compaq NonStop Kernel, the value depends on whether SNAX or ICE is used; see Chapter 20, “Setting up communication in Compaq NonStop Kernel” on page 307.

**If the transmission protocol is NetBIOS**
This is the NetBIOS name defined on the remote machine.

**If the transmission protocol is SPX**
This is an SPX-style address consisting of a 4-byte network address, a 6-byte node address and a 2-byte socket number. Enter these in hexadecimal, with the network and node addresses separated by a fullstop and the socket number in brackets. For example:

```
CONNAME('0a0b0c0d.804abcde23a1(5e86)')
```

If the socket number is omitted, the default WebSphere MQ SPX socket number is used. The default is X'5E86'.

**Note:** The definition of transmission protocol is contained in “Transport type (TRPTYPE)” on page 99.

### Convert message (CONVERT)
Application message data is usually converted by the receiving application. However, if the remote queue manager is on a platform that does not support data conversion, use this channel attribute to specify that the message should be converted into the format required by the receiving system before transmission.

This attribute applies only to sender, cluster-sender, server, and cluster-receiver channels and does not apply to WebSphere MQ for z/OS with CICS.

The possible values are ‘yes’ and ‘no’. If you specify ‘yes’, the application data in the message is converted before sending if you have specified one of the built-in format names, or a data conversion exit is available for a user-defined format (See...
Convert message (CONVERT)

The WebSphere MQ Application Programming Guide. If you specify ‘no’, the application data in the message is not converted before sending.

Description (DESCR)

This contains up to 64 bytes of text that describes the channel definition.

Note: The maximum number of characters is reduced if the system is using a double byte character set (DBCS).

Use characters from the character set identified by the coded character set identifier (CCSID) for the queue manager to ensure that the text is translated correctly if it is sent to another queue manager.

Disconnect interval (DISCINT)

This is a time-out attribute, specified in seconds, for the server, cluster-sender, sender, and cluster-receiver channels. The interval is measured from the point at which a batch ends, that is when the batch size is reached or when the batch interval expires and the transmission queue becomes empty. If no messages arrive on the transmission queue during the specified time interval, the channel closes down. (The time is approximate.)

The close-down exchange of control data between the two ends of the channel includes an indication of the reason for closing. This ensures that the corresponding end of the channel remains available to start up again.

On all platforms except z/OS with CICS, you can specify any number of seconds from zero through 999 999 where a value of zero means no disconnect; wait indefinitely.

In z/OS using CICS, you can specify any number of seconds from zero through 9999 where a value of zero means disconnect as soon as the transmission queue is empty.

Note: Performance is affected by the value specified for the disconnect interval.

A very low value (a few seconds) may cause excessive overhead in constantly starting up the channel. A very large value (more than an hour) could mean that system resources are unnecessarily held up. For WebSphere MQ for AIX, iSeries, HP-UX, Linux, Solaris, and Windows systems, and MQSeries V5.1 for Compaq Tru64 UNIX, and OS/2 Warp, and for WebSphere MQ for z/OS without CICS, you can also specify a heartbeat interval, so that when there are no messages on the transmission queue, the sending MCA will send a heartbeat flow to the receiving MCA, thus giving the receiving MCA an opportunity to quiesce the channel without waiting for the disconnect interval to expire. For these two values to work together effectively, the heartbeat interval value must be significantly lower than the disconnect interval value.

A value for the disconnect interval of a few minutes is a reasonable value to use. Change this value only if you understand the implications for performance, and you need a different value for the requirements of the traffic flowing down your channels.

For more information, see “Stopping and quiescing channels” on page 67.
Heartbeat interval (HBINT)

This attribute applies to WebSphere MQ for AIX, iSeries, HP-UX, Linux, Solaris, and Windows systems, and MQSeries V5.1 for Compaq Tru64 UNIX, and OS/2 Warp, and for WebSphere MQ for z/OS without CICS. You can specify the approximate time between heartbeat flows that are to be passed from a sending MCA when there are no messages on the transmission queue. Heartbeat flows unblock the receiving MCA, which is waiting for messages to arrive or for the disconnect interval to expire. When the receiving MCA is unblocked it can disconnect the channel without waiting for the disconnect interval to expire. Heartbeat flows also free any storage buffers that have been allocated for large messages and close any queues that have been left open at the receiving end of the channel.

The value is in seconds and must be in the range 0 through 999 999. A value of zero means that no heartbeat flows are to be sent. The default value is 300. To be most useful, the value should be significantly less than the disconnect interval value.

This attribute is valid for sender, cluster-sender, server, receiver, cluster-receiver, and requester channels. Other than on z/OS and iSeries, it also applies to server-connection and client-connection channels. On these channels, heartbeats flow when a server MCA has issued an MQGET command with the WAIT option on behalf of a client application.

KeepAlive Interval (KAINT)

The KeepAlive Interval parameter is used to specify a time-out value for a channel.

The KeepAlive Interval parameter is a value passed to the communications stack specifying the KeepAlive timing for the channel. It allows you to specify a different keepalive value for each channel.

The value indicates a time, in seconds, and must be in the range 0 to 99999. A KeepAlive value of 0 indicates that channel KeepAlive is not enabled for the channel. When the KeepAlive Interval is set to 0, keepalive still occurs if a value has been specified for TCP/IP keepalive. On z/OS, this occurs when the TCPKEEP=YES parameter is specified in CSQ6CHIP. On other platforms, it occurs when the KEEPALIVE=YES parameter is specified in the TCP stanza in the distributed queuing configuration file.

You can also set KAINT to a value of AUTO. If KAINT is set to AUTO, the keepalive value is based on the value of the negotiated HBINT as follows:

<table>
<thead>
<tr>
<th>Negotiated HBINT</th>
<th>KAINT</th>
</tr>
</thead>
<tbody>
<tr>
<td>&gt;0</td>
<td>Negotiated HBINT + 60 seconds</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

This parameter is valid for all channel types. The value is ignored for all channels that have a TransportType (TRPTYPE) other than TCP or SPX

KAINT is only available on WebSphere MQ for z/OS.
Local Address (LOCLADDR)

This parameter specifies the local communications address for the channel. When a LOCLADDR value is specified, a channel that is stopped and then restarted continues to use the TCP/IP address specified in LOCLADDR. In recovery scenarios, this could be useful when the channel is communicating through a firewall, because it removes problems caused by the channel restarting with a different IP address, specified by the TCP/IP stack to which it is connected.

This parameter is valid for the following channel types:

- Sender
- Server
- Requester
- Client-connection
- Cluster-receiver
- Cluster-sender

The value used is the optional IP address and optional port or port range to be used for outbound TCP/IP communications. The format is as follows:

LOCLADDR([ip-addr][([low-port[,high-port]])])

where "ip-addr" is specified in dotted alphanumeric or decimal form, for example, (MACH1.ABC.COM) or (19.22.11.162), and "low-port" and "high-port" are port numbers enclosed in parentheses. When two port values are specified, the channel binds to the address specified, using an available port within the range covered by the two port values. All values are optional.

The maximum length of the string is MQ_LOCAL_ADDRESS_LENGTH.

Note: If the LOCLADDR port is in use, TCP/IP requires a time period to release the previously used port. If enough time is not left, and if only 1 LOCLADDR port is specified, the previously used port will not be available and so a random port will be chosen rather than the LOCLADDR port.

Long retry count (LONGRTY)

Specify the maximum number of times that the channel is to try allocating a session to its partner. If the initial allocation attempt fails, the short retry count number is decremented and the channel retries the remaining number of times. If it still fails, it retries a long retry count number of times with an interval of long retry interval between each try. If it is still unsuccessful, the channel closes down. The channel must subsequently be restarted with a command (it is not started automatically by the channel initiator).

(Retry is not attempted if the cause of failure is such that a retry is not likely to be successful.)

If the channel initiator or queue manager stops while the channel is retrying, the short retry count and long retry count are reset when the channel initiator or queue manager is restarted.

The long retry count attribute is valid only for channel types of sender, cluster-sender, server, and cluster-receiver. It is also valid for requester channels on
Long retry count (LONGRTY)

z/OS if you are using CICS. It may be set from zero through 999 999 999. On z/OS using CICS, it may be set from zero through 999, and the long and short retries have the same count.

Note: For OS/2, iSeries, UNIX systems, and Windows systems, in order for retry to be attempted a channel initiator must be running. The channel initiator must be monitoring the initiation queue specified in the transmission queue that the channel is using.

Long retry interval (LONGTMR)

The approximate interval in seconds that the channel is to wait before retrying to establish connection, during the long retry mode.

The interval between retries may be extended if the channel has to wait to become active.

The channel tries to connect long retry count number of times at this long interval, after trying the short retry count number of times at the short retry interval.

This is valid only for channel types of sender, cluster-sender, server, and cluster-receiver. It is also valid for requester channels on z/OS if you are using CICS. It may be set from zero through 999 999. On z/OS using CICS, it may be set from zero through 999.

LU 6.2 mode name (MODENAME)

This is for use with LU 6.2 connections. It gives extra definition for the session characteristics of the connection when a communication session allocation is performed.

When using side information for SNA communications, the mode name is defined in the CPI-C Communications Side Object or APPC side information and this attribute should be left blank; otherwise, it should be set to the SNA mode name.

The name must be one to eight alphanumeric characters long.

It is not valid for receiver or server-connection channels.

LU 6.2 transaction program name (TPNAME)

This is for use with LU 6.2 connections. It is the name, or generic name, of the transaction program (MCA) to be run at the far end of the link.

When using side information for SNA communications, the transaction program name is defined in the CPI-C Communications Side Object or APPC side information and this attribute should be left blank. Otherwise, this name is required by sender channels and requester channels except on Z/OS using CICS where it is required in the channel definition but is ignored unless the server is initiating the conversation.

On platforms other than Compaq NonStop Kernel, the name can be up to 64 characters long. See Chapter 20, “Setting up communication in Compaq NonStop Kernel” on page 307 for more information about that platform.

If the remote system is WebSphere MQ for z/OS using CICS, the transaction is:
LU 6.2 transaction program name (TPNAME)

- CKRC when you are defining a sender channel, or a server channel that acts as a sender
- CKSV when you are defining a requester channel of a requester-server pair
- CKRC when you are defining a requester channel of a requester-sender pair

On other platforms, this should be set to the SNA transaction program name, unless the CONNAME contains a side-object name in which case it should be set to blanks. The actual name is taken instead from the CPI-C Communications Side Object, or the APPC side information data set.

This information is set in a different way on other platforms; see the section in this book about setting up communication for your platform.

Maximum message length (MAXMSGL)

Specifies the maximum length of a message that can be transmitted on the channel.

On WebSphere MQ for AIX, iSeries, HP-UX, Linux, Solaris, and Windows systems, and MQSeries V5.1 for OS/2 Warp, and VSE/ESA, specify a value greater than or equal to zero, and less than or equal to the maximum message length for the queue manager. See the MAXMSGL parameter of the ALTER QMGR command in the WebSphere MQ Script (MQSC) Command Reference book for more information.

On other platforms, specify a value greater than or equal to zero, and less than or equal to 4 194 304 bytes. On WebSphere MQ for z/OS, specify a value greater than or equal to zero, and less than or equal to 104 857 600 bytes.

Because various implementations of WebSphere MQ systems exist on different platforms, the size available for message processing may be limited in some applications. This number must reflect a size that your system can handle without stress. When a channel starts up, the lower of the two numbers at each end of the channel is taken.

Notes:
1. If splitting of messages is not supported at either end of a channel, the maximum message size cannot be greater than the negotiated maximum transmission size.
2. The IBM WebSphere MQ products that this edition of the book applies to all support message splitting. Other WebSphere MQ products do not support message splitting.
3. For a comparison of the functions available, including the different maximum message lengths available see the WebSphere MQ Application Programming Guide.
4. You may use a maximum message size of 0 which will be taken to mean that the size is to be set to the local queue manager maximum value.

Maximum transmission size

If you are using CICS for distributed queuing on z/OS, you can specify the maximum transmission size, in bytes, that your channel is allowed to use when transmitting a message, or part of a message. When a channel starts up, this value is negotiated between the sending and receiving channels and the lower of the two values is agreed. The maximum size is 32 000 bytes on TCP/IP, but the maximum usable size is 32 000 bytes less the message descriptor. On VSE/ESA, the maximum size is 64 000 bytes on SNA.

Use this facility to ensure that system resources are not exceeded by your channels. Set this value in conjunction with the maximum message size, remembering to
allow for message descriptors. An error situation may be created if the message size is allowed to exceed the transmission size, and message splitting is not supported.

**Notes:**
1. If channel startup negotiation results in a size less than the minimum required for the local channel program, no messages can be transferred.

**Message channel agent name (MCANAME)**
This attribute is reserved and should not be used.

**Message channel agent type (MCATYPE)**
For WebSphere MQ for AIX, iSeries, HP-UX, Linux, Solaris, and Windows systems, and MQSeries V5.1 for OS/2 Warp, this attribute may be specified as a ‘process’ or a ‘thread’. This parameter is valid for channel types of sender, cluster-sender (on WebSphere MQ for AIX, iSeries, HP-UX, Linux, Solaris, and Windows systems, and MQSeries V5.1 for Compaq Tru64 UNIX, and OS/2 Warp), server, requester, or cluster-receiver. On WebSphere MQ for z/OS, it is supported only for channels with a channel type of cluster-receiver. The MCA type is used when the channel is started locally to determine how the channel is run.

Advantages of running as a process include:
- Isolation for each channel providing greater integrity
- Job authority specific for each channel
- Control over job scheduling

Advantages of threads include:
- Much reduced use of storage
- Easier configuration by typing on the command line
- Faster execution - it is quicker to start a thread than to instruct the operating system to start a process

For channel types of sender, server, and requester, the default is ‘process’. For channel types of cluster-sender and cluster-receiver, the default is ‘thread’. These defaults can change during your installation.

If you specify ‘process’ on the channel definition, a RUNMQCHL process is started. If you specify ‘thread’, the MCA runs on a thread of the RUNMQCHI process. On the machine that receives the inbound allocates, the MCA runs as a thread or process depending on whether you use inetd or RUNMQLSR.

**Message channel agent user identifier (MCAUSER)**
This is not valid for z/OS using CICS; it is not valid for channels of client-connection type.

This attribute is the user identifier (a string) to be used by the MCA for authorization to access WebSphere MQ resources, including (if PUT authority is DEF) authorization to put the message to the destination queue for receiver or requester channels.

On WebSphere MQ for Windows, the user identifier may be domain-qualified by using the format, user@domain, where the domain must be either the Windows systems domain of the local system or a trusted domain.
Message exit name (MSGEXIT)

If this attribute is blank, the MCA uses its default user identifier.

Message exit name (MSGEXIT)

Specifies the name of the user exit program to be run by the channel message exit.

On AIX, Compaq Tru64 UNIX, HP-UX, iSeries, Linux, OS/2 Warp, Solaris, Windows systems, and z/OS this can be a list of names of programs that are to be run in succession. Leave blank, if no channel message exit is in effect.

The format and maximum length of this attribute depend on the platform, as for "Receive exit name (RCVEXIT)" on page 95.

The message exit is not supported on client-connection or server-connection channels.

Message exit user data (MSGDATA)

Specifies user data that is passed to the channel message exits.

In WebSphere MQ for AIX, iSeries, HP-UX, Linux, Solaris, Windows systems, and z/OS, and MQSeries V5.1 for Compaq Tru64 UNIX, and OS/2 Warp, you can run a sequence of message exits. The limitations on the user data length and an example of how to specify MSGDATA for more than one exit are as shown for RCVDATA. See "Receive exit user data (RCVDATA)" on page 95.

On other platforms the maximum length of the string is 32 characters.

Message-retry exit name (MREXIT)

Specifies the name of the user exit program to be run by the message-retry user exit. Leave blank if no message-retry exit program is in effect.

The format and maximum length of the name depend on the platform, as for "Receive exit name (RCVEXIT)" on page 95.

This parameter is only valid for receiver, cluster-receiver, and requester channels. It is not supported on WebSphere MQ for z/OS.

Message-retry exit user data (MRDATA)

This is passed to the channel message-retry exit when it is called.

This parameter is only valid for receiver, cluster-receiver, and requester channels. It is not supported on WebSphere MQ for z/OS.

Message retry count (MRRTY)

This is the number of times the channel will retry before it decides it cannot deliver the message.

This attribute controls the action of the MCA only if the message-retry exit name is blank. If the exit name is not blank, the value of MRRTY is passed to the exit for the exit’s use, but the number of retries performed (if any) is controlled by the exit, and not by this attribute.

The value must be in the range 0 to 999 999 999. A value of zero means that no retries will be performed.
Message retry count (MRRTY)

This parameter is only valid for receiver, cluster-receiver, and requester channels. It is not supported on WebSphere MQ for z/OS.

Message retry interval (MRTMR)

This is the minimum interval of time that must pass before the channel can retry the MQPUT operation. This time interval is in milliseconds.

This attribute controls the action of the MCA only if the message-retry exit name is blank. If the exit name is not blank, the value of MRTMR is passed to the exit for the exit’s use, but the retry interval is controlled by the exit, and not by this attribute.

The value must be in the range 0 to 999,999,999. A value of zero means that the retry will be performed as soon as possible (provided that the value of MRRTY is greater than zero).

This parameter is only valid for receiver, cluster-receiver, and requester channels. It is not supported on WebSphere MQ for z/OS.

Network-connection priority (NETPRTY)

The priority for the network connection. Distributed queuing will choose the path with the highest priority if there are multiple paths available. The value must be in the range 0 through 9; 0 is the lowest priority.

This parameter is valid only for cluster-receiver channels.

This parameter is valid only on AIX, HP-UX, Linux, OS/2 Warp, z/OS without CICS, iSeries, Solaris, and Windows systems.

Nonpersistent message speed (NPMSPEED)

For WebSphere MQ for AIX, HP-UX, iSeries, Solaris and Windows systems, WebSphere MQ for z/OS without CICS, and MQSeries V5.1 for Compaq Tru64 UNIX and OS/2 Warp, you can specify the speed at which nonpersistent messages are to be sent. You can specify either ‘normal’ or ‘fast’. The default is ‘fast’, which means that nonpersistent messages on a channel are not transferred within transactions. The advantage of this is that nonpersistent messages become available for retrieval far more quickly. The disadvantage is that because they are not part of a transaction, messages may be lost if there is a transmission failure or if the channel stops when the messages are in transit. See “Fast, nonpersistent messages” on page 22.

This attribute is valid for sender, cluster-sender, server, receiver, cluster-receiver, and requester channels.

Password (PASSWORD)

You can specify a password of maximum length 12 characters, although only the first 10 characters are used.

The password may be used by the MCA when attempting to initiate a secure LU 6.2 session with a remote MCA. It is valid for channel types of sender, server, requester, or client-connection.

This does not apply to WebSphere MQ for z/OS except for client-connection channels.
PUT authority (PUTAUT)

Use this attribute to choose the type of security processing to be carried out by the MCA when executing:

- An MQPUT command to the destination queue (for message channels), or
- An MQI call (for MQI channels).

You can choose one of the following:

Process security, also called default authority (DEF)
The default user ID is used.

On platforms, with Process security, you choose to have the queue security based on the user ID that the process is running under. The user ID is that of the process, or user, running the MCA at the receiving end of the message channel.

The queues are opened with this user ID and the open option MQOO_SET_ALL_CONTEXT.

Context security (CTX)
The alternate user ID is used from the context information associated with the message.

The UserIdentifier in the message descriptor is moved into the AlternateUserId field in the object descriptor. The queue is opened with the open options MQOO_SET_ALL_CONTEXT and MQOO_ALTERNATE_USER_AUTHORITY.

Only Message Channel Agent security (ONLYMCA)
The default user ID is used.

On platforms, with ONLYMCA security, you choose to have the queue security based on the user ID that the process is running under. The user ID is that of the process, or user, running the MCA at the receiving end of the message channel.

The queues are opened with this user ID and the open option MQOO_SET_ALL_CONTEXT.

Alternate Message Channel Agent security (ALTMCA)
This is the same as for ONLYMCA security but allows you to use context.

This parameter is only valid for receiver, requester, cluster-receiver, and server-connection channels. Context security and alternate message channel agent security values are not supported on server-connection channels.

Further details about:

- Context fields and open options can be found in the WebSphere MQ Application Programming Guide.
- Security can be found in the WebSphere MQ System Administration Guide for WebSphere MQ for AIX, iSeries, HP-UX, Linux, Solaris, and Windows systems, and MQSeries V5.1 for Compaq Tru64 UNIX, and OS/2 Warp, or in the MQSeries System Administration book for your platform.

Note: On WebSphere MQ for z/OS it is possible for two userids to be checked. Specific details of userids used by the channel initiator on z/OS can be found in the WebSphere MQ for z/OS System Setup Guide.
Queue manager name (QMNAME)

This applies to a channel of client-connection type only. It is the name of the queue manager or queue manager group to which a WebSphere MQ client application can request connection.

Receive exit name (RCVEXIT)

Specifies the name of the user exit program to be run by the channel receive user exit. In WebSphere MQ for AIX, iSeries, HP-UX, Linux, Solaris, Windows systems, and z/OS, and MQSeries V5.1 for Compaq Tru64 UNIX, and OS/2 Warp this can be a list of names of programs that are to be run in succession. Leave blank, if no channel receive user exit is in effect.

The format and maximum length of this attribute depend on the platform:
- On z/OS it is a load module name, maximum length 8 characters, except for client-connection channels where the maximum length is 128 characters.
- On iSeries it is of the form:
  
  libname/progname

  when specified in CL commands.

  When specified in WebSphere MQ Commands (MQSC) it has the form:

  progrname libname

  where progrname occupies the first 10 characters, and libname the second 10 characters (both blank-padded to the right if necessary). The maximum length of the string is 20 characters.
- On OS/2 and Windows it is of the form:

  dllname(functionname)

  where dllname is specified without the suffix “.DLL”. The maximum length of the string is 40 characters.
- On UNIX systems, Compaq OpenVMS Alpha, and Compaq NonStop Kernel it is of the form:

  libraryname(functionname)

  The maximum length of the string is 40 characters.

On AIX, Compaq Tru64 UNIX, HP-UX, Linux, iSeries, OS/2 Warp, Solaris, Windows systems, and z/OS, you can specify a list of receive, send, or message exit program names. The names should be separated by a comma, a space, or both. For example:

RCVEXIT(exit1 exit2)
MSGEXIT(exit1, exit2)
SENDEXIT(exit1, exit2)

In WebSphere MQ for AIX, iSeries, HP-UX, Linux, Solaris, Windows systems, and z/OS, and MQSeries V5.1 for Compaq Tru64 UNIX, and OS/2 Warp, the total length of the string of exit names and strings of user data for a particular type of exit is limited to 500 characters. In WebSphere MQ for iSeries you can list up to 10 exit names. In WebSphere MQ for z/OS you can list up to eight exit names.

Receive exit user data (RCVDATA)

Specifies user data that is passed to the receive exit.
Receive exit user data (RCVDATA)

In WebSphere MQ for AIX, iSeries, HP-UX, Linux, Solaris, z/OS, and Windows systems, and MQSeries V5.1 for Compaq Tru64 UNIX, and OS/2 Warp you can run a sequence of receive exits. The string of user data for a series of exits should be separated by a comma, spaces, or both. For example:

RCVDATA(exit1_data exit2_data)
MSGDATA(exit1_data,exit2_data)
SENDDATA(exit1_data, exit2_data)

In WebSphere MQ for AIX, iSeries, HP-UX, Linux, Solaris, and Windows systems, and MQSeries V5.1 for Compaq Tru64 UNIX, and OS/2 Warp, the length of the string of exit names and strings of user data is limited to 500 characters. In WebSphere MQ for iSeries you can specify up to 10 exit names and the length of user data for each is limited to 32 characters. In WebSphere MQ for z/OS you can specify up to eight strings of user data each of length 32 characters.

On other platforms the maximum length of the string is 32 characters.

Security exit name (SCYEXIT)

Specifies the name of the exit program to be run by the channel security exit. Leave blank if no channel security exit is in effect.

The format and maximum length of the name depend on the platform, as for “Receive exit name (RCVEXIT)” on page 95

Security exit user data (SCYDATA)

Specifies user data that is passed to the security exit. The maximum length is 32 characters.

Send exit name (SENDEXIT)

Specifies the name of the exit program to be run by the channel send exit. In WebSphere MQ for AIX, iSeries, HP-UX, Linux, Solaris, z/OS, and Windows systems, and MQSeries V5.1 for Compaq Tru64 UNIX, and OS/2 Warp, this can be a list of names of programs that are to be run in sequence. Leave blank if no channel send exit is in effect.

The format and maximum length of this attribute depend on the platform, as for “Receive exit name (RCVEXIT)” on page 95

Send exit user data (SENDDATA)

Specifies user data that is passed to the send exit.

In WebSphere MQ for AIX, iSeries, HP-UX, Linux, Solaris, z/OS, and Windows systems, and MQSeries V5.1 for Compaq Tru64 UNIX, and OS/2 Warp, you can run a sequence of send exits. The limitations on the user data length and an example of how to specify SENDDATA for more than one exit, are as shown for RCVDATA. See “Receive exit user data (RCVDATA)” on page 95

On other platforms the maximum length of the string is 32 characters.

Sequence number wrap (SEQWRAP)

This is the highest number the message sequence number reaches before it restarts at 1. In z/OS using CICS, this number is of interest only when sequential delivery of messages is selected. It is not valid for channel types of client-connection or server-connection.
The value of the number should be high enough to avoid a number being reissued while it is still being used by an earlier message. The two ends of a channel must have the same sequence number wrap value when a channel starts up; otherwise, an error occurs.

The value may be set from 100 through 999 999 999 (1 through 9 999 999 for z/OS using CICS).

**Sequential delivery**

This applies only to z/OS using CICS. Set this to ‘YES’ when using sequential numbering of messages. If one side of the channel requests this facility, it must be accepted by the other side.

There could be a performance penalty associated with the use of this option.

For other platforms, the MCA always uses message sequence numbering.

**Short retry count (SHORTRTTY)**

Specify the maximum number of times that the channel is to try allocating a session to its partner. If the initial allocation attempt fails, the short retry count is decremented and the channel retries the remaining number of times with an interval, defined in the short retry interval attribute, between each attempt. If it still fails, it retries long retry count number of times with an interval of long retry interval between each attempt. If it is still unsuccessful, the channel terminates.

(Retry is not attempted if the cause of failure is such that a retry is not likely to be successful.)

If the channel initiator or queue manager stops while the channel is retrying, the short retry count and long retry count are reset when the channel initiator or queue manager is restarted.

The short retry count attribute is valid only for channel types of sender, cluster-sender, server, and cluster-receiver. It is also valid for requester channels on z/OS if you are using CICS. It may be set from zero through 999 999 999 (1 through 999 for z/OS using CICS, and the long and short retries have the same count).

**Note:** On OS/2 Warp, iSeries, UNIX systems, and Windows systems, in order for retry to be attempted a channel initiator must be running. The channel initiator must be monitoring the initiation queue specified in the transmission queue that the channel in using.

**Short retry interval (SHORTTMR)**

Specify the approximate interval in seconds that the channel is to wait before retrying to establish connection, during the short retry mode.

The interval between retries may be extended if the channel has to wait to become active.
Short retry interval (SHORTTMR)

This attribute is valid only for channel types of sender, cluster-sender, server, and cluster-receiver. It is also valid for requester channels on z/OS if you are using CICS. It may be set from zero through 999 999. (0 through 999 for z/OS using CICS).

SSL Cipher Specification (SSLCIPH)

SSLCIPH defines a single CipherSpec for an SSL connection. Both ends of a WebSphere MQ SSL channel definition must include the parameter and the SSLCIPH values must specify the same CipherSpec on both ends of the channel.

The value is a string with a maximum length of 32 characters.

SSLCIPH is valid for all channel types. It is supported on AIX, HP-UX, Linux, iSeries, Solaris, Windows, and z/OS. It is valid only for channels with a transport type (TRPTYPE) of TCP. If the TRPTYPE is not TCP, the data is ignored and no error message is issued.

SSLCIPH is an optional parameter.

For more information on SSLCIPH, see WebSphere MQ Script (MQSC) Command Reference and WebSphere MQ Security.

SSL Client Authentication (SSLCAUTH)

SSLCAUTH is used to define whether the channel needs to receive and authenticate an SSL certificate from an SSL client.

This parameter is valid on all channel types that can ever receive a channel initiation flow, except for sender channels. It is valid for receiver, server-connection, cluster-receiver, server, and requester channels.

The value of SSLCAUTH can be set to OPTIONAL or REQUIRED. The default value is REQUIRED. If SSLCAUTH is set to OPTIONAL, and the peer SSL client sends a certificate, the certificate is processed as normal.

You can specify a value for SSLCAUTH on a non-SSL channel definition, one on which SSLCIPH is missing or blank. You can use this to temporarily disable SSL for debugging without first having to clear and then reinput the SSL parameters.

This parameter is valid on AIX, HP-UX, Linux, iSeries, Solaris, Windows, and z/OS

SSLCAUTH is an optional parameter.

For more information on SSLCAUTH, see WebSphere MQ Script (MQSC) Command Reference and WebSphere MQ Security.

SSL Peer (SSLPEER)

The SSLPEER parameter is used to check the Distinguished Name (DN) of the certificate from the peer queue manager or client at the other end of a WebSphere MQ channel. If the DN received from the peer does not match the SSLPEER value, the channel does not start.

SSLPEER is an optional parameter. If a value is not specified, the peer DN is not checked when the channel is started.
This parameter is valid for all channel types.

This parameter is supported only on AIX, HP-UX, Linux, OS/2 Warp, iSeries, Solaris, Windows, and z/OS.

On z/OS the maximum length of the parameter is 256 bytes. On all other platforms it is 1024 bytes.

On z/OS the parameter values used are not checked. If you input incorrect values, the channel fails at startup, and error messages are written to the error log at both ends of the channel. A Channel SSL Error event is also generated at both ends of the channel. On platforms that support SSLPEER, other than z/OS, the validity of the string is checked when it is first input.

You can specify a value for SSLPEER on a non-SSL channel definition, one on which SSLCIPH is missing or blank. You can use this to temporarily disable SSL for debugging without having to clear and later reinput the SSL parameters.

For more information on using SSLPEER, see WebSphere MQ Script (MQSC) Command Reference and WebSphere MQ Security.

**Target system identifier**

This is for z/OS using CICS only. It identifies the particular CICS system where the sending or requesting channel transaction is to run.

The default is blank, which means the CICS system where you are logged on. The name may be one through four alphanumeric characters.

**Transaction identifier**

This only applies to z/OS using CICS.

The name of the local CICS transaction that you want to start. If you do not specify a value, the name of the supplied transaction for the channel type is used.

**Transmission queue name (XMITQ)**

The name of the transmission queue from which messages are retrieved. This is required for channels of type sender or server, it is not valid for other channel types.

Provide the name of the transmission queue to be associated with this sender or server channel, that corresponds to the queue manager at the far side of the channel. The transmission queue may be given the same name as the queue manager at the remote end.

**Transport type (TRPTYPE)**

This does not apply to z/OS using CICS.
Transport type Transmission protocol (TRPTYPE)

The possible values are:

<table>
<thead>
<tr>
<th>Transport Type</th>
<th>Transmission Protocol</th>
</tr>
</thead>
<tbody>
<tr>
<td>LU62</td>
<td>LU 6.2</td>
</tr>
<tr>
<td>TCP</td>
<td>TCP/IP</td>
</tr>
<tr>
<td>UDP</td>
<td>UDP (1)</td>
</tr>
<tr>
<td>NETBIOS</td>
<td>NetBIOS (2)</td>
</tr>
<tr>
<td>SPX</td>
<td>SPX (2)</td>
</tr>
</tbody>
</table>

Notes:
1. UDP is supported on WebSphere MQ for AIX only.
2. For use on OS/2 and Windows. Can also be used on z/OS for defining client-connection channels for use on OS/2 and Windows.

User ID (USERID)

On WebSphere MQ for AIX, iSeries, HP-UX, Linux, Solaris, and Windows systems, and MQSeries V5.1 for Compaq Tru64 UNIX, and OS/2 Warp, you can specify a task user identifier of 20 characters. On other platforms, you can specify a task user identifier of maximum length 12 characters, although only the first 10 characters are used.

The user ID may be used by the MCA when attempting to initiate a secure SNA session with a remote MCA. It is valid for channel types of sender, server, requester, or client-connection.

This does not apply to WebSphere MQ for z/OS, except for client-connection channels.

On the receiving end, if passwords are kept in encrypted format and the LU 6.2 software is using a different encryption method, an attempt to start the channel fails with invalid security details. You can avoid this by modifying the receiving SNA configuration to either:

- Turn off password substitution, or
- Define a security user ID and password.

On the receiving end, if passwords are kept in encrypted format and the LU 6.2 software is using a different encryption method, an attempt to start the channel fails with invalid security details. You can avoid this by modifying the receiving SNA configuration to either:

- Turn off password substitution, or
- Define a security user ID and password.
Chapter 7. Example configuration chapters in this book

Throughout the following parts of the book, there is a series of chapters containing examples of how to configure the various platforms to communicate with each other. These chapters describe tasks performed to establish a working WebSphere MQ network. The tasks were to establish WebSphere MQ sender and receiver channels to enable bi-directional message flow between the platforms over all supported protocols.

Figure 32 is a conceptual representation of a single channel and the WebSphere MQ objects associated with it.

This is a simple example, intended to introduce only the basic elements of the WebSphere MQ network. It does not demonstrate the use of triggering which is described in “Triggering channels” on page 20.

The objects in this network are:
- A remote queue
- A transmission queue
- A local queue
- A sender channel
- A receiver channel

Appl1 and Appl2 are both application programs; Appl1 is putting messages and Appl2 is receiving them.

Appl1 puts messages to a remote queue. The definition for this remote queue specifies the name of a target queue manager, a local queue on that queue manager, and a transmission queue on this the local queue manager.

When the queue manager receives the request from Appl1 to put a message to the remote queue, it looks at the queue definition and sees that the destination is remote. It therefore puts the message, along with a transmission header, straight
Example configurations

onto the transmission queue specified in the definition. The message remains on
the transmission queue until the channel becomes available, which may happen
immediately.

A sender channel has in its definition a reference to one, and one only,
transmission queue. When a channel is started, and at other times during its
normal operation, it will look at this transmission queue and send any messages
on it to the target system. The message has in its transmission header details of the
destination queue and queue manager.

The intercommunication examples in the following chapters describe in detail the
creation of each of the objects described above, for a variety of platform
combinations.

On the target queue manager, definitions are required for the local queue and the
receiver side of the channel. These objects operate independently of each other and
so can be created in any sequence.

On the local queue manager, definitions are required for the remote queue, the
transmission queue, and the sender side of the channel. Since both the remote
queue definition and the channel definition refer to the transmission queue name,
it is advisable to create the transmission queue first.

Network infrastructure

The configuration examples assume that all the systems are connected to a Token
Ring network with the exception of z/OS and VSE/ESA, which communicate via a
3745 (or equivalent) that is attached to the Token Ring, and Solaris, which is on an
adjacent local area network (LAN) also attached to the 3745.

It is also assumed that, for SNA, all the required definitions in VTAM and network
control program (NCP) are in place and activated for the LAN-attached platforms
to communicate over the wide area network (WAN).

Similarly, for TCP, it is assumed that nameserver function is available, either via a
domain nameserver or via locally held tables (for example a host file).

Communications software

Working configurations are given in the following chapters for the following
network software products:

• SNA
  – Communications Manager/2 Version 1.11
  – Communications Server for Windows systems, Version 5.0
  – AIX Communications Server, V5.0
  – Hewlett-Packard SNAplus2
  – AT&T GIS SNA Services Version 2.06 or later
  – OS/400® Version 4 Release 4
  – SunLink Peer-to-Peer Version 9.1
  – OS/390 Version 2 Release 4
  – CICS/VSE® Version 2 Release 1
Communications software

- TCP
  - TCP for OS/2 Version 2
  - Microsoft® Windows NT® Version 4 or later, Windows 2000
  - AIX Version 4 Release 1.4
  - HP-UX Version 10.2 or later
  - AT&T GIS UNIX Release 2.03.01
  - Sun Solaris Release 2.4 or later
  - OS/400 Version 4 Release 4
  - TCP for z/OS
  - Digital UNIX Version 4.0 or later
- NetBIOS
- SPX
- UDP

How to use the communication examples

The following chapters contain example configurations:

- Chapter 11, “Example configuration - MQSeries for OS/2 Warp” on page 143
- Chapter 12, “Example configuration - IBM WebSphere MQ for Windows” on page 167
- Chapter 14, “Example configuration - IBM WebSphere MQ for AIX” on page 197
- Chapter 15, “Example configuration - IBM MQSeries for Compaq Tru64 UNIX” on page 215
- Chapter 16, “Example configuration - IBM WebSphere MQ for HP-UX” on page 219
- Chapter 17, “Example configuration - IBM MQSeries for AT&T GIS UNIX, V2.2” on page 243
- Chapter 18, “Example configuration - IBM WebSphere MQ for Solaris” on page 257
- Chapter 26, “Example configuration - IBM WebSphere MQ for z/OS” on page 365
- Chapter 42, “Example configuration - MQSeries for VSE/ESA” on page 549
- Chapter 31, “Example configuration - IBM WebSphere MQ for z/OS using CICS” on page 421
- Chapter 35, “Example configuration - WebSphere MQ for z/OS using queue-sharing groups” on page 447

The information in the example-configuration chapters describes the tasks that were carried out on a single platform, to set up communication to another of the platforms, and then describes the WebSphere MQ tasks to establish a working channel to that platform. Wherever possible, the intention is to make the information as generic as possible. Thus, to connect any two WebSphere MQ queue managers on different platforms, you should need to refer to only the relevant two chapters. Any deviations or special cases are highlighted as such. Of course, you can also connect two queue managers running on the same platform (on different machines or on the same machine). In this case, all the information can be derived from the one chapter.

The examples only cover how to set up communications where clustering is not being used. For information about setting up communications while using clustering, see the [WebSphere MQ Queue Manager Clusters] book. The communications’ configuration values given here still apply.
Using communication examples

Each chapter contains a worksheet in which you can find the parameters used in the example configurations. There is a short description of each parameter and some guidance on where to find the equivalent values in your system. When you have a set of values of your own, record these in the spaces on the worksheet. As you proceed through the chapter, you will find cross-references to these values as you need them.

Notes:
1. Example queue manager names usually reflect the platform that the queue manager runs on, but MVS™ is used for both z/OS and MVS/ESA, which are essentially the same.
2. The sequence number wrap value for sender definitions defaults to 999999999 for Version 2 WebSphere MQ products.
3. For connections to WebSphere MQ for z/OS the examples, in general, cover only connection without using CICS. See Chapter 29, “Preparing WebSphere MQ for z/OS when using CICS” on page 413 for information about connecting using CICS.

IT responsibilities

Because the IT infrastructure can vary greatly between organizations, it is difficult to indicate who, within an organization, controls and maintains the information required to complete each parameter value. To understand the terminology used in the following chapters, consider the following guidelines as a starting point.

- **System administrator** is used to describe the person (or group of people) who installs and configures the software for a specific platform.
- **Network administrator** is used to describe the person who controls LAN connectivity, LAN address assignments, network naming conventions, and so on. This person may be in a separate group or may be part of the system administration group.

In most z/OS installations, there is a group responsible for updating the ACF/VTAM®, ACF/NCP, and TCP/IP software to support the network configuration. The people in this group should be the main source of information needed when connecting any WebSphere MQ platform to WebSphere MQ for z/OS. They may also influence or mandate network naming conventions on LANs and you should verify their span of control before creating your definitions.

- **A specific type of administrator, for example CICS administrator** is indicated in cases where we can more clearly describe the responsibilities of the person.

The example-configuration chapters do not attempt to indicate who is responsible for and able to set each parameter. In general, several different people may be involved.
Part 3. DQM in WebSphere MQ for UNIX systems, and Windows systems, and MQSeries for Compaq OpenVMS Alpha, Compaq NonStop Kernel, and OS/2 Warp

Chapter 8. Monitoring and controlling channels on distributed platforms

The DQM channel control function

Functions available

Getting started with objects

Creating objects

Creating default objects

How are default objects created?

Changing the default objects

Creating a channel

Create channel example

Displaying a channel

Display channel examples

Displaying channel status

Display channel status examples

Starting a channel

Renaming a channel

Channel attributes and channel types

Channel functions

Create

Change

Delete

Display

Display Status

Ping

Ping with LU 6.2

Start

Stop

Stop quiesce example

Stop terminate example

Reset

Resolve

Chapter 9. Preparing WebSphere MQ for distributed platforms

Transmission queues and triggering

Creating a transmission queue

Triggering channels

Example definitions for triggering

Examples for WebSphere MQ for AIX, iSeries, HP-UX, Linux, Solaris, and Windows systems, and MQSeries V5.1 for Compaq Tru64 UNIX, and OS/2 Warp

Starting the channel initiator

Stopping the channel initiator

Chapter 10. Setting up communication for OS/2 and Windows

Deciding on a connection

Defining a TCP connection

Sending end

Receiving on TCP

Using the TCP/IP listener

Using the TCP listener backlog option

Using the WebSphere MQ listener

Using the TCP/IP SO_KEEPALIVE option

Defining an LU 6.2 connection

Sending end for OS/2

Sending end for Windows

Receiving on LU 6.2

Using the RUNMQLSR command

Using Communications Manager/2 on OS/2

Using Microsoft SNA Server on Windows

Defining a NetBIOS connection

Defining the WebSphere MQ local NetBIOS name

Establishing the queue manager NetBIOS session, command, and name limits

Establishing the LAN adapter number

Initiating the connection

Target listener

Defining an SPX connection

Sending end

Using the SPX KEEPALIVE option (OS/2 only)

Receiving on SPX

Using the SPX listener backlog option

Using the WebSphere MQ listener

IPX/SPX parameters

OS/2

DOS and Windows 3.1 client

Windows systems

Windows 95 and Windows 98

Chapter 11. Example configuration - MQSeries for OS/2 Warp

Configuration parameters for an LU 6.2 connection

Configuration worksheet

Explanation of terms

Establishing an LU 6.2 connection

Defining local node characteristics

On Windows systems

User IDs across systems

User IDs on OS/2

System extensions and user-exit programs

Running channels and listeners as trusted applications

What next?
### DQM in distributed platforms

<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Configuring a DLC</td>
<td>149</td>
</tr>
<tr>
<td>Configuring the local node</td>
<td>149</td>
</tr>
<tr>
<td>Adding a local LU</td>
<td>150</td>
</tr>
<tr>
<td>Adding a transaction program definition</td>
<td>150</td>
</tr>
<tr>
<td>Configuring a mode</td>
<td>151</td>
</tr>
<tr>
<td>Connecting to a peer system</td>
<td>151</td>
</tr>
<tr>
<td>Adding a peer connection</td>
<td>152</td>
</tr>
<tr>
<td>Defining a partner LU</td>
<td>152</td>
</tr>
<tr>
<td>Connecting to a host system</td>
<td>153</td>
</tr>
<tr>
<td>Adding a host connection</td>
<td>153</td>
</tr>
<tr>
<td>Defining a partner LU</td>
<td>154</td>
</tr>
<tr>
<td>Verifying the configuration</td>
<td>154</td>
</tr>
<tr>
<td>What next?</td>
<td>155</td>
</tr>
<tr>
<td>Establishing a TCP connection</td>
<td>180</td>
</tr>
<tr>
<td>Establishing an LU 6.2 connection</td>
<td>172</td>
</tr>
<tr>
<td>Establishing a SPX connection</td>
<td>181</td>
</tr>
<tr>
<td>Establishing a NetBIOS connection</td>
<td>180</td>
</tr>
<tr>
<td>Establishing an SPX connection</td>
<td>181</td>
</tr>
</tbody>
</table>

### WebSphere MQ Intercommunication

<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Configuration parameters for an LU 6.2 connection</td>
<td>163</td>
</tr>
<tr>
<td>Configuration parameters using SNA</td>
<td>163</td>
</tr>
<tr>
<td>Configuration parameters using TCP</td>
<td>163</td>
</tr>
<tr>
<td>Configuration parameters using TCP/IP</td>
<td>164</td>
</tr>
<tr>
<td>Configuration parameters using NetBIOS</td>
<td>164</td>
</tr>
<tr>
<td>Configuration parameters using IPX/SPX</td>
<td>164</td>
</tr>
<tr>
<td>Configuration parameters using SPX</td>
<td>164</td>
</tr>
<tr>
<td>Configuration parameters using NetBIOS</td>
<td>164</td>
</tr>
<tr>
<td>Configuration parameters using IPX/SPX</td>
<td>164</td>
</tr>
<tr>
<td>Configuration parameters using TCP/IP</td>
<td>164</td>
</tr>
<tr>
<td>Configuration parameters using SNA</td>
<td>164</td>
</tr>
</tbody>
</table>

### Chapter 13. Setting up communication in UNIX systems

<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Deciding on a connection</td>
<td>191</td>
</tr>
<tr>
<td>Defining a TCP connection</td>
<td>191</td>
</tr>
<tr>
<td>Sending end</td>
<td>191</td>
</tr>
<tr>
<td>Receiving on TCP</td>
<td>192</td>
</tr>
<tr>
<td>Sending end</td>
<td>192</td>
</tr>
<tr>
<td>Using the TCP listener</td>
<td>193</td>
</tr>
<tr>
<td>Using the TCP listener backlog option</td>
<td>193</td>
</tr>
<tr>
<td>Using the WebSphere MQ listener</td>
<td>193</td>
</tr>
<tr>
<td>Using the TCP/IP SO_KEEPALIVE option</td>
<td>194</td>
</tr>
<tr>
<td>Defining an LU 6.2 connection</td>
<td>194</td>
</tr>
<tr>
<td>Sending end</td>
<td>195</td>
</tr>
<tr>
<td>Receiving on LU 6.2</td>
<td>195</td>
</tr>
</tbody>
</table>

### Chapter 14. Example configuration - IBM

<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>WebSphere MQ for Windows configuration</td>
<td>183</td>
</tr>
<tr>
<td>WebSphere MQ for Windows configuration</td>
<td>183</td>
</tr>
<tr>
<td>Default configuration</td>
<td>183</td>
</tr>
<tr>
<td>Basic configuration</td>
<td>183</td>
</tr>
<tr>
<td>Channel configuration</td>
<td>184</td>
</tr>
<tr>
<td>WebSphere MQ for Windows sender-channel definitions using SNA</td>
<td>186</td>
</tr>
<tr>
<td>WebSphere MQ for Windows receiver-channel definitions using SNA</td>
<td>187</td>
</tr>
<tr>
<td>WebSphere MQ for Windows TCP/IP definitions using TCP/IP</td>
<td>187</td>
</tr>
<tr>
<td>WebSphere MQ for Windows NetBIOS definitions using NetBIOS</td>
<td>187</td>
</tr>
<tr>
<td>WebSphere MQ for Windows SPX definitions using SPX</td>
<td>188</td>
</tr>
<tr>
<td>WebSphere MQ for Windows IPX definitions using IPX/SPX</td>
<td>188</td>
</tr>
<tr>
<td>WebSphere MQ for Windows SNA definitions using SNA</td>
<td>187</td>
</tr>
<tr>
<td>WebSphere MQ for Windows receiver-channel definitions using IPX/SPX</td>
<td>187</td>
</tr>
<tr>
<td>Automatic startup</td>
<td>188</td>
</tr>
<tr>
<td>Running channels as processes or threads</td>
<td>188</td>
</tr>
<tr>
<td>Multiple thread support — pipelining</td>
<td>189</td>
</tr>
<tr>
<td>Channel exit considerations</td>
<td>189</td>
</tr>
</tbody>
</table>

---

WebSphere MQ for AIX configuration

Chapter 12. Example configuration - IBM

<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>MQSeries for OS/2 Warp configuration</td>
<td>159</td>
</tr>
<tr>
<td>Basic configuration</td>
<td>160</td>
</tr>
<tr>
<td>Channel configuration</td>
<td>160</td>
</tr>
<tr>
<td>MQSeries for OS/2 Warp sender-channel definitions using SNA</td>
<td>163</td>
</tr>
<tr>
<td>MQSeries for OS/2 Warp receiver-channel definitions using SNA</td>
<td>163</td>
</tr>
<tr>
<td>MQSeries for OS/2 Warp sender-channel definitions using TCP</td>
<td>163</td>
</tr>
<tr>
<td>MQSeries for OS/2 Warp receiver-channel definitions using TCP/IP</td>
<td>164</td>
</tr>
<tr>
<td>MQSeries for OS/2 Warp receiver-channel definitions using NetBIOS</td>
<td>164</td>
</tr>
<tr>
<td>MQSeries for OS/2 Warp receiver-channel definitions using IPX/SPX</td>
<td>164</td>
</tr>
<tr>
<td>MQSeries for OS/2 Warp receiver-channel definitions using IPX/SPX</td>
<td>164</td>
</tr>
<tr>
<td>Running channels as processes or threads</td>
<td>165</td>
</tr>
</tbody>
</table>

Chapter 14. Example configuration - IBM

<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Configuration parameters for an LU 6.2 connection</td>
<td>197</td>
</tr>
<tr>
<td>Configuration worksheet</td>
<td>197</td>
</tr>
<tr>
<td>Explanation of terms</td>
<td>200</td>
</tr>
<tr>
<td>Server for AIX V5</td>
<td>203</td>
</tr>
<tr>
<td>Configuring your node</td>
<td>203</td>
</tr>
<tr>
<td>Configuring connectivity to the network</td>
<td>204</td>
</tr>
<tr>
<td>Defining a local LU</td>
<td>206</td>
</tr>
<tr>
<td>Defining a transaction program</td>
<td>207</td>
</tr>
<tr>
<td>Establishing a TCP connection</td>
<td>208</td>
</tr>
<tr>
<td>What next?</td>
<td>209</td>
</tr>
<tr>
<td>Establishing a UDP connection</td>
<td>209</td>
</tr>
<tr>
<td>What next?</td>
<td>209</td>
</tr>
</tbody>
</table>

WebSphere MQ for AIX configuration

Basic configuration                                      | 210  |
<p>| Channel configuration                                      | 210  |</p>
<table>
<thead>
<tr>
<th>Chapter 15. Example configuration - IBM</th>
<th>MQSeries for Compaq Tru64 UNIX</th>
<th>215</th>
</tr>
</thead>
<tbody>
<tr>
<td>Establishing a TCP connection</td>
<td></td>
<td>215</td>
</tr>
<tr>
<td>What next?</td>
<td></td>
<td>215</td>
</tr>
<tr>
<td>MQSeries for Compaq Tru64 UNIX</td>
<td></td>
<td>215</td>
</tr>
<tr>
<td>Basic configuration</td>
<td></td>
<td>216</td>
</tr>
<tr>
<td>Channel configuration</td>
<td></td>
<td>216</td>
</tr>
<tr>
<td>MQSeries for Compaq Tru64 UNIX</td>
<td></td>
<td>218</td>
</tr>
<tr>
<td>sender-channel definitions using TCP/</td>
<td></td>
<td>218</td>
</tr>
<tr>
<td>IP</td>
<td></td>
<td>218</td>
</tr>
<tr>
<td>MQSeries for Compaq Tru64 UNIX</td>
<td></td>
<td>218</td>
</tr>
<tr>
<td>receiver-channel definitions using TCP/IP</td>
<td></td>
<td>218</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Chapter 16. Example configuration - IBM</th>
<th>WebSphere MQ for HP-UX</th>
<th>219</th>
</tr>
</thead>
<tbody>
<tr>
<td>Configuration parameters for an LU 6.2 connection</td>
<td>219</td>
<td></td>
</tr>
<tr>
<td>Configuration worksheet</td>
<td>219</td>
<td></td>
</tr>
<tr>
<td>Explanation of terms</td>
<td>222</td>
<td></td>
</tr>
<tr>
<td>Establishing a session using HP SNAplus2</td>
<td>223</td>
<td></td>
</tr>
<tr>
<td>SNAplus2 configuration</td>
<td>223</td>
<td></td>
</tr>
<tr>
<td>Defining a local node</td>
<td>225</td>
<td></td>
</tr>
<tr>
<td>Adding a Token Ring Port</td>
<td>225</td>
<td></td>
</tr>
<tr>
<td>Defining a local LU</td>
<td>226</td>
<td></td>
</tr>
<tr>
<td>APPC configuration</td>
<td>227</td>
<td></td>
</tr>
<tr>
<td>Defining a remote node</td>
<td>227</td>
<td></td>
</tr>
<tr>
<td>Defining a partner LU</td>
<td>228</td>
<td></td>
</tr>
<tr>
<td>Defining a link station</td>
<td>229</td>
<td></td>
</tr>
<tr>
<td>Defining a mode</td>
<td>231</td>
<td></td>
</tr>
<tr>
<td>Adding CPI-C information</td>
<td>233</td>
<td></td>
</tr>
<tr>
<td>Adding a TP definition using HP SNAplus2</td>
<td>235</td>
<td></td>
</tr>
<tr>
<td>Release 5</td>
<td>235</td>
<td></td>
</tr>
<tr>
<td>Adding a TP definition using HP SNAplus2</td>
<td>235</td>
<td></td>
</tr>
<tr>
<td>Release 6</td>
<td>235</td>
<td></td>
</tr>
<tr>
<td>HP-UX operation</td>
<td>237</td>
<td></td>
</tr>
<tr>
<td>What next?</td>
<td>237</td>
<td></td>
</tr>
<tr>
<td>Establishing a TCP connection</td>
<td>237</td>
<td></td>
</tr>
<tr>
<td>What next?</td>
<td>238</td>
<td></td>
</tr>
<tr>
<td>WebSphere MQ for HP-UX configuration</td>
<td>238</td>
<td></td>
</tr>
<tr>
<td>Basic configuration</td>
<td>238</td>
<td></td>
</tr>
<tr>
<td>Channel configuration</td>
<td>238</td>
<td></td>
</tr>
<tr>
<td>WebSphere MQ for HP-UX sender-channel</td>
<td>241</td>
<td></td>
</tr>
<tr>
<td>definitions using SNA</td>
<td>241</td>
<td></td>
</tr>
<tr>
<td>WebSphere MQ for HP-UX receiver-channel definitions using SNA</td>
<td>241</td>
<td></td>
</tr>
<tr>
<td>WebSphere MQ for HP-UX invokable TP setup</td>
<td>241</td>
<td></td>
</tr>
<tr>
<td>WebSphere MQ for HP-UX sender-channel</td>
<td>241</td>
<td></td>
</tr>
<tr>
<td>definitions using TCP</td>
<td>241</td>
<td></td>
</tr>
</tbody>
</table>

Part 3. DQM in WebSphere MQ for UNIX systems, and Windows systems, and MQSeries for Compaq OpenVMS Alpha, Compaq NonStop Kernel, and OS/2 Warp 107

<table>
<thead>
<tr>
<th>Chapter 18. Example configuration - IBM</th>
<th>WebSphere MQ for Solaris</th>
<th>257</th>
</tr>
</thead>
<tbody>
<tr>
<td>Loading the WebSphere MQ library to support SNA</td>
<td>257</td>
<td></td>
</tr>
<tr>
<td>Configuration parameters for an LU 6.2 connection using SunLink Version 9.1</td>
<td>258</td>
<td></td>
</tr>
<tr>
<td>Configuration worksheet</td>
<td>258</td>
<td></td>
</tr>
<tr>
<td>Explanation of terms</td>
<td>261</td>
<td></td>
</tr>
<tr>
<td>Establishing a connection using SunLink Version 9.1</td>
<td>262</td>
<td></td>
</tr>
<tr>
<td>SunLink 9.1 base configuration</td>
<td>262</td>
<td></td>
</tr>
<tr>
<td>Configuring a PU 2.1 server</td>
<td>262</td>
<td></td>
</tr>
<tr>
<td>Adding a LAN connection</td>
<td>263</td>
<td></td>
</tr>
<tr>
<td>Configuring a connection to a remote PU</td>
<td>265</td>
<td></td>
</tr>
<tr>
<td>Configuring an independent LU</td>
<td>266</td>
<td></td>
</tr>
<tr>
<td>Configuring a partner LU</td>
<td>268</td>
<td></td>
</tr>
<tr>
<td>Configuring the session mode</td>
<td>269</td>
<td></td>
</tr>
<tr>
<td>Configuring a transaction program</td>
<td>270</td>
<td></td>
</tr>
<tr>
<td>Invokable TP path</td>
<td>271</td>
<td></td>
</tr>
<tr>
<td>CPI-C side information</td>
<td>271</td>
<td></td>
</tr>
<tr>
<td>What next?</td>
<td>272</td>
<td></td>
</tr>
<tr>
<td>Configuration parameters for an LU 6.2 connection using SNAP-IX</td>
<td>272</td>
<td></td>
</tr>
<tr>
<td>Configuration worksheet</td>
<td>272</td>
<td></td>
</tr>
<tr>
<td>Explanation of terms</td>
<td>274</td>
<td></td>
</tr>
<tr>
<td>Establishing a session using SNAP-IX</td>
<td>276</td>
<td></td>
</tr>
<tr>
<td>SNAP-IX configuration</td>
<td>276</td>
<td></td>
</tr>
<tr>
<td>Defining a local node</td>
<td>277</td>
<td></td>
</tr>
</tbody>
</table>
DQM in distributed platforms

Adding a Token Ring Port ........................................... 278
Defining a local LU .................................................. 279
APPC configuration .................................................... 279
Defining a remote node ............................................... 279
Defining a partner LU .................................................. 280
Defining a link station .................................................. 280
Defining a mode .......................................................... 282
Adding CPI-C information .............................................. 283
Adding a TP definition using SNAP-IX Release 6 ............... 286
SNAP-IX operation ....................................................... 288
What next? ................................................................. 288
Establishing a TCP connection ......................................... 288
What next? ................................................................. 289
WebSphere MQ for Solaris configuration ............................. 293
Basic configuration ...................................................... 289
Channel configuration ................................................... 289
WebSphere MQ for Solaris sender-channel definitions using SNA using SunLink Version 9.1 ............................ 292
WebSphere MQ for Solaris sender-channel definitions using SNAP-IX SNA .................................................. 292
WebSphere MQ for Solaris receiver-channel definitions using SNA .......................................................... 292
WebSphere MQ for Solaris sender-channel definitions using TCP .......................................................... 293
WebSphere MQ for Solaris receiver-channel definitions using TCP/IP .................................................. 293

Chapter 19. Setting up communication in Compaq OpenVMS Alpha systems ........................................... 295
Deciding on a connection ............................................... 295
Defining a TCP connection .............................................. 296
Sending end ................................................................. 296
Receiving channels using Compaq (DIGITAL) TCP/IP services (UCX) for OpenVMS .................................................. 296
Using the TCP/IP SO_KEEPALIVE option .............................................. 297
Receiving channels using Cisco MultiNet for OpenVMS .................................................. 297
Receiving channels using Attachmate PathWay for OpenVMS .................................................. 298
Receiving channels using Process Software Corporation TCPware .................................................. 298
Defining an LU 6.2 connection ........................................... 299
SNA configuration ......................................................... 299
Defining access names .................................................... 300
Specifying SNA configuration parameters to WebSphere MQ .................................................. 301
Passing parameters to sender and requester channel pairs .................................................. 301
Running senders and requesters ........................................... 301
Passing parameters to servers and receivers .............................................. 302
Running servers and receivers .............................................. 302
Ending the SNA Listener process ........................................... 302
Sample WebSphere MQ configuration .................................... 302
Problem solving .......................................................... 303
Defining a DECnet Phase IV connection .................................... 304
Sending end ................................................................. 304
Receiving on DECnet Phase IV ............................................. 304
Defining a DECnet Phase V connection .................................... 304

Chapter 20. Setting up communication in Compaq NonStop Kernel .................................................. 307
Deciding on a connection ............................................... 307
SNA channels ............................................................. 307
LU 6.2 responder processes .............................................. 309
TCP channels ............................................................. 309
Communications examples .............................................. 309
SNAX communications example ....................................... 309
SCF SNA line configuration file .............................................. 309
SYSGEN parameters ........................................................ 312
SNAX/APC process configuration ............................................ 313
Channel definitions ........................................................ 317
ICE communications example ............................................. 317
Configuring the ICE process ............................................. 317
Defining the line and APC information ...................................... 318
Channel definitions for ICE .................................................. 320
TCP/IP communications example ......................................... 321
TCPConfig stanza in QMINI ................................................. 321
Defining a TCP sender channel .............................................. 321
Defining a TCP receiver channel .............................................. 321
Defining a TCP/IP sender channel on the remote system .............. 322

Chapter 21. Message channel planning example for distributed platforms ........................................... 323
What the example shows .................................................. 323
Queue manager QM1 example .............................................. 325
Queue manager QM2 example .............................................. 326
Running the example ...................................................... 327
Expanding this example .................................................. 327

Chapter 22. Example S/NIX and DC/OSx configuration files .................................................. 329
Configuration file on bight ................................................. 330
Configuration file on forties ................................................. 331
Working configuration files for Pyramid DC/OSx .............................................. 332
Output of dbd command .................................................... 332
Chapter 8. Monitoring and controlling channels on distributed platforms

For DQM you need to create, monitor, and control the channels to remote queue managers. You can use the following types of command to do this:

The WebSphere MQ commands (MQSC)
You can use the MQSC as single commands in an MQSC session in OS/2, Windows NT, Compaq OpenVMS Alpha, Compaq NonStop Kernel, and UNIX systems. To issue more complicated, or multiple, commands the MQSC can be built into a file that you then run from the command line. For full details see the WebSphere MQ Script (MQSC) Command Reference book. This chapter gives some simple examples of using MQSC for distributed queuing.

Control commands
You can also issue control commands at the command line for some of these functions. Reference material for these commands is contained in the WebSphere MQ System Administration Guide book for WebSphere MQ for AIX, iSeries, HP-UX, Linux, Solaris, and Windows systems, and MQSeries V5.1 for Compaq Tru64 UNIX, and OS/2 Warp.

Programmable command format commands
See the WebSphere MQ Programmable Command Formats and Administration Interface book for information about using these commands.

Message Queue Management facility
On Compaq NonStop Kernel, you can use the Message Management facility. See the MQSeries for Compaq NonStop Kernel System Administration for information about this facility.

IBM WebSphere MQ Explorer
On Windows, you can use an MMC snap-in called the WebSphere MQ Explorer. This provides a graphical administration interface to perform administrative tasks as an alternative to using control commands or MQSC commands.

Each queue manager has a DQM component for controlling interconnections to compatible remote queue managers.

For a list of the functions available to you when setting up and controlling message channels, using the two types of commands, see Table 9 on page 110.

The DQM channel control function

The channel control function provides the interface and function for administration and control of message channels between systems.

It consists of commands, programs, a file for the channel definitions, and a storage area for synchronization information. The following is a brief description of the components.

• The channel commands are a subset of the WebSphere MQ Commands (MQSC).
• You use MQSC and the control commands to:
  – Create, copy, display, change, and delete channel definitions
Channel control function

- Start and stop channels, ping, reset channel sequence numbers, and resolve in-doubt messages when links cannot be re-established
- Display status information about channels
- The channel definition file (CDF), amqrfda.dat:
  - Is indexed on channel name
  - Holds channel definitions
- A storage area holds sequence numbers and logical unit of work (LUW) identifiers. These are used for channel synchronization purposes.

Functions available

Table 9 shows the full list of WebSphere MQ functions that you may need when setting up and controlling channels. The channel functions are explained in this chapter.

For more details of the control commands that you issue at the command line, see the WebSphere MQ System Administration Guide book for WebSphere MQ for AIX, iSeries, HP-UX, Linux, Solaris, and Windows systems, and MQSeries V5.1 for Compaq Tru64 UNIX, and OS/2 Warp.

The MQSC commands are fully described in the WebSphere MQ Script (MQSC) Command Reference book.

Table 9. Functions available in OS/2, Windows systems, Compaq OpenVMS Alpha, Compaq NonStop Kernel, and UNIX systems

<table>
<thead>
<tr>
<th>Function</th>
<th>MQSC</th>
<th>WebSphere MQ Explorer equivalent?</th>
<th>WebSphere MQ Service snap-in equivalent?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Change queue manager</td>
<td>ALTER QMGR</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Create queue manager</td>
<td>crtmqm</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Delete queue manager</td>
<td>dltmqm</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Display queue manager</td>
<td>DISPLAY QMGR</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>End queue manager</td>
<td>endmqm</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Ping queue manager</td>
<td>PING QMGR</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Start queue manager</td>
<td>strmqm</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Add a queue manager to Windows systems Service Control Manager</td>
<td></td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Display command server</td>
<td>dspmqcsv</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>End command server</td>
<td>endmqcsv</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Start command server</td>
<td>strmqcsv</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Change queue</td>
<td>ALTER QALIAS ALTER QLOCAL ALTER QMODEL ALTER QREMOTE</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Clear queue</td>
<td>CLEAR QLOCAL CLEAR QUEUE</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Function</td>
<td>Control commands</td>
<td>MQSC</td>
<td>WebSphere MQ Explorer equivalent?</td>
</tr>
<tr>
<td>-------------------------</td>
<td>------------------</td>
<td>-------------------------------</td>
<td>----------------------------------</td>
</tr>
<tr>
<td>Create queue</td>
<td>DEFINE QALIAS</td>
<td>DEFINE QLOCAL</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>DEFINE QMODEL</td>
<td>DEFINE QREMOTE</td>
<td></td>
</tr>
<tr>
<td>Delete queue</td>
<td>DELETE QALIAS</td>
<td>DELETE QLOCAL</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>DELETE QMODEL</td>
<td>DELETE QREMOTE</td>
<td></td>
</tr>
<tr>
<td>Display queue</td>
<td>DISPLAY QUEUE</td>
<td></td>
<td>Yes</td>
</tr>
<tr>
<td>Process functions</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Change process</td>
<td>ALTER PROCESS</td>
<td></td>
<td>Yes</td>
</tr>
<tr>
<td>Create process</td>
<td>DEFINE PROCESS</td>
<td></td>
<td>Yes</td>
</tr>
<tr>
<td>Delete process</td>
<td>DELETE PROCESS</td>
<td></td>
<td>Yes</td>
</tr>
<tr>
<td>Display process</td>
<td>DISPLAY PROCESS</td>
<td></td>
<td>Yes</td>
</tr>
<tr>
<td>Channel functions</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Change channel</td>
<td>ALTER CHANNEL</td>
<td></td>
<td>Yes</td>
</tr>
<tr>
<td>Create channel</td>
<td>DEFINE CHANNEL</td>
<td></td>
<td>Yes</td>
</tr>
<tr>
<td>Delete channel</td>
<td>DELETE CHANNEL</td>
<td></td>
<td>Yes</td>
</tr>
<tr>
<td>Display channel</td>
<td>DISPLAY CHANNEL</td>
<td></td>
<td>Yes</td>
</tr>
<tr>
<td>Display channel status</td>
<td>DISPLAY CHSTATUS</td>
<td></td>
<td>Yes</td>
</tr>
<tr>
<td>End channel</td>
<td>STOP CHANNEL</td>
<td></td>
<td>Yes</td>
</tr>
<tr>
<td>Ping channel</td>
<td>PING CHANNEL</td>
<td></td>
<td>Yes</td>
</tr>
<tr>
<td>Reset channel</td>
<td>RESET CHANNEL</td>
<td></td>
<td>Yes</td>
</tr>
<tr>
<td>Resolve channel</td>
<td>RESOLVE CHANNEL</td>
<td></td>
<td>Yes</td>
</tr>
<tr>
<td>Run channel</td>
<td>runmqchl</td>
<td>START CHANNEL</td>
<td>Yes</td>
</tr>
<tr>
<td>Run channel initiator</td>
<td>runmqchi</td>
<td>START CHINIT (not Compaq NonStop Kernel)</td>
<td>No</td>
</tr>
</tbody>
</table>
Functions available

Table 9. Functions available in OS/2, Windows systems, Compaq OpenVMS Alpha, Compaq NonStop Kernel, and UNIX systems (continued)

<table>
<thead>
<tr>
<th>Function</th>
<th>Control commands</th>
<th>MQSC</th>
<th>WebSphere MQ Explorer equivalent?</th>
<th>WebSphere MQ Service snap-in equivalent?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Run listener</td>
<td>runmqlsr (not AT&amp;T GIS UNIX and Compaq Tru64 UNIX)</td>
<td>START LISTENER (not Compaq NonStop Kernel)</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>End listener</td>
<td>endmqlsr (OS/2, Windows systems, AIX, HP-UX, Solaris, and SINIX and DC/0Sx only)</td>
<td>No</td>
<td>Yes</td>
<td></td>
</tr>
</tbody>
</table>

Getting started with objects

Use the WebSphere MQ commands (MQSC) or the WebSphere MQ Explorer on Windows systems to:
1. Define message channels and associated objects
2. Monitor and control message channels

The objects you may need to define are:
- Transmission queues
- Remote queue definitions
- Queue manager alias definitions
- Reply-to queue alias definitions
- Reply-to local queues
- Processes for triggering (MCAs)
- Message channel definitions

Channels must be completely defined, and their associated objects must exist and be available for use, before a channel can be started. This chapter shows you how to do this.

In addition, the particular communication link for each channel must be defined and available before a channel can be run. For a description of how LU 6.2, TCP/IP, NetBIOS, SPX, and DECnet links are defined, see the particular communication guide for your installation. See also the example configuration chapters in this book.

Creating objects

Use MQSC to create the queue and alias objects: transmission queues, remote queue definitions, queue manager alias definitions, reply-to queue alias definitions, and reply-to local queues.

Also create the definitions of processes for triggering (MCAs) in a similar way.

For an example showing how to create all the required objects see Chapter 21, “Message channel planning example for distributed platforms” on page 323.
Creating default objects

In WebSphere MQ for AIX, iSeries, HP-UX, Linux, Solaris, and Windows systems, and MQSeries V5.1 for Compaq Tru64 UNIX, and OS/2 Warp, default objects are created automatically when a queue manager is created. These objects are queues, channels, a process definition, and administration queues. They correspond to the objects that are created when you run the amqscoma.tst sample command file on earlier releases of these products and on other WebSphere MQ products.

How are default objects created?

When you use the CRTMQM command to create a queue manager, the command also initiates a program to create a set of default objects.

1. Each default object is created in turn. The program keeps a count of how many objects are successfully defined, how many already existed and were replaced, and how many unsuccessful attempts there were.

2. The program displays the results to you and if any errors occurred, directs you to the appropriate error log for details.

When the program has finished running, you can use the STRMQM command to start the queue manager.

See the WebSphere MQ System Administration Guide book for information about the CRTMQM and STRMQM commands and a list of default objects.

Changing the default objects

Once the default objects have been created, you can replace them at any time by running the STRMQM command with the -c option. When you specify the -c option, the queue manager is started temporarily while the objects are created and is then shut down again. You must use the STRMQM command again, without the -c option, if you want to start the queue manager.

If you wish to make any changes to the default objects, you can create your own version of the old amqscoma.tst file and edit it.

Creating a channel

To create a new channel you have to create two channel definitions, one at each end of the connection. You create the first channel definition at the first queue manager. Then you create the second channel definition at the second queue manager, on the other end of the link.

Both ends must be defined using the same channel name. The two ends must have compatible channel types, for example: Sender and Receiver.

To create a channel definition for one end of the link use the MQSC command DEFINE CHANNEL. Include the name of the channel, the channel type for this end of the connection, a connection name, a description (if required), the name of the transmission queue (if required), and the transmission protocol. Also include any other attributes that you want to be different from the system default values for the required channel type, using the information you have gathered previously.

You are provided with help in deciding on the values of the channel attributes in Chapter 6, “Channel attributes” on page 77

Note: You are very strongly recommended to name all the channels in your network uniquely. Including the source and target queue manager names in the channel name is a good way to do this.
Create channel example

DEFINE CHANNEL(QM1.TO.QM2) CHLTYPE(SDR) +
DESCR('Sender channel to QM2') +
CONNAME(QM2) TRPTYPE(TCP) XMITQ(QM2) CONVERT(YES)

In all the examples of MQSC the command is shown as it would appear in a file of commands, and as it would be typed in OS/2, Windows NT, UNIX systems, Compaq OpenVMS Alpha, or Compaq NonStop Kernel. The two methods look identical, except that to issue a command interactivity, you must first start an MQSC session. Type runmqsc, for the default queue manager, or runmqsc qmname where QMNAME is the name of the required queue manager. Then type any number of commands, as shown in the examples.

For portability, you should restrict the line length of your commands to 72 characters. Use a concatenation character as shown to continue over more than one line. On Compaq NonStop Kernel, use Ctrl-y to end the input at the command line, or enter exit or quit. On OS/2, Windows NT, or Compaq OpenVMS Alpha use Ctrl-z. On UNIX systems, use Ctrl-d. Alternatively, on V5.3 of WebSphere MQ for AIX, Compaq Tru64 UNIX, HP-UX, OS/2 Warp, Solaris, and Windows, use the end command.

Displaying a channel

Use the MQSC command DISPLAY CHANNEL, specifying the channel name, the channel type (optional), and the attributes you want to see, or specifying that all attributes are to be displayed. In WebSphere MQ for AIX, iSeries, HP-UX, Linux, Solaris, and Windows systems, and MQSeries V5.1 for Compaq Tru64 UNIX, and OS/2 Warp the ALL parameter of the DISPLAY CHANNEL command is assumed by default if no specific attributes are requested and the channel name specified is not generic.

The attributes are described in Chapter 6, “Channel attributes” on page 77.

Display channel examples

DISPLAY CHANNEL(QM1.TO.QM2) TRPTYPE,CONVERT
DISPLAY CHANNEL(QM1.TO.*) TRPTYPE,CONVERT
DISPLAY CHANNEL(*) TRPTYPE,CONVERT
DISPLAY CHANNEL(QM1.TO.QMR34) ALL

Displaying channel status

Use the MQSC command DISPLAY CHSTATUS, specifying the channel name and whether you want the current status of channels or the status of saved information.

Display channel status examples

DISPLAY CHSTATUS(*) CURRENT
DISPLAY CHSTATUS(QM1.TO.*) SAVED

Note that the saved status does not apply until at least one batch of messages has been transmitted on the channel. In WebSphere MQ for AIX, iSeries, HP-UX, Linux, Solaris, and Windows systems, and MQSeries V5.1 for Compaq Tru64 UNIX, and OS/2 Warp status is also saved when a channel is stopped (using the STOP CHL command) and when the queue manager is ended.
Starting a channel

For applications to be able to exchange messages you must start a listener program for inbound connections (or, in the case of UNIX systems, create a listener attachment). In OS/2, Windows systems, and Compaq NonStop Kernel, use the runmqlsr command to start the WebSphere MQ listener process. Any inbound requests for channel attachment start MCAs as threads of this listener process. In Compaq OpenVMS Alpha, each receiver or server channel requires a listener process that then starts a channel process.

```
runmqlsr -t tcp -m QM2
```

For outbound connections you must start the channel in one of the following three ways:

1. Use the MQSC command START CHANNEL, specifying the channel name, to start the channel as a process or a thread, depending on the MCATYPE parameter. (If channels are started as threads, they are threads of a channel initiator.)
   
   ```
   START CHANNEL(QM1.TO.QM2)
   ```

2. Use the control command runmqchl to start the channel as a process.
   
   ```
   runmqchl -c QM1.TO.QM2 -m QM1
   ```

3. Use the channel initiator to trigger the channel.

Renaming a channel

To rename a message channel, use MQSC to carry out the following steps:

1. Use STOP CHANNEL to stop the channel.
2. Use DEFINE CHANNEL to create a duplicate channel definition with the new name.
3. Use DISPLAY CHANNEL to check that it has been created correctly.
4. Use DELETE CHANNEL to delete the original channel definition.

If you decide to rename a message channel, remember that a channel has two channel definitions, one at each end. Make sure you rename the channel at both ends at the same time.

Channel attributes and channel types

The channel attributes for each type of channel are shown in Table 7 on page 77. The channel attributes are described in detail in Chapter 6, “Channel attributes” on page 77. Client-connection channels and server-connection channels are described in the WebSphere MQ Clients book.

Channel functions

The channel functions available are shown in Table 9 on page 110. Here some more detail is given about the channel functions.

Create

You can create a new channel definition using the default values supplied by WebSphere MQ, specifying the name of the channel, the type of channel you are creating, the communication method to be used, the transmission queue name and the connection name.
Channel functions

The channel name must be the same at both ends of the channel, and unique within the network. However, you should restrict the characters used to those that are valid for WebSphere MQ object names.

Change

Use the MQSC command ALTER CHANNEL to change an existing channel definition, except for the channel name, or channel type.

Delete

Use the MQSC command DELETE CHANNEL to delete a named channel.

Display

Use the MQSC command DISPLAY CHANNEL to display the current definition for the channel.

Display Status

The MQSC command DISPLAY CHSTATUS displays the status of a channel whether the channel is active or inactive. It applies to all message channels. It does not apply to MQI channels other than server-connection channels on WebSphere MQ for AIX, iSeries, HP-UX, Linux, Solaris, and Windows systems, and MQSeries V5.1 for Compaq Tru64 UNIX, and OS/2 Warp. See “Displaying channel status” on page 114.

Information displayed includes:
- Channel name
- Communication connection name
- In-doubt status of channel (where appropriate)
- Last sequence number
- Transmission queue name (where appropriate)
- The in-doubt identifier (where appropriate)
- The last committed sequence number
- Logical unit of work identifier
- Process ID
- Thread ID (OS/2 and Windows only)

Ping

Use the MQSC command PING CHANNEL to exchange a fixed data message with the remote end. This gives some confidence to the system supervisor that the link is available and functioning.

Ping does not involve the use of transmission queues and target queues. It uses channel definitions, the related communication link, and the network setup. It can only be used if the channel is not currently active.

It is available from sender and server channels only. The corresponding channel is started at the far side of the link, and performs the startup parameter negotiation. Errors are notified normally.

The result of the message exchange is presented as Ping complete or an error message.

Ping with LU 6.2

When Ping is invoked, by default no USERID or password flows to the receiving end. If USERID and password are required, they can be created at the initiating
end in the channel definition. If a password is entered into the channel definition, it is encrypted by WebSphere MQ before being saved. It is then decrypted before flowing across the conversation.

**Start**

Use the MQSC command START CHANNEL for sender, server, and requester channels. It should not be necessary where a channel has been set up with queue manager triggering.

Also use the START CHANNEL command for receiver channels that have a disabled status, and on WebSphere MQ for AIX, iSeries, HP-UX, Linux, Solaris, and Windows systems, and MQSeries V5.1 for Compaq Tru64 UNIX, and OS/2 Warp, for server-connection channels that have a disabled status. Starting a receiver or server-connection channel that is in disabled status resets the channel and allows it to be started from the remote channel.

When started, the sending MCA reads the channel definition file and opens the transmission queue. A channel start-up sequence is executed, which remotely starts the corresponding MCA of the receiver or server channel. When they have been started, the sender and server processes await messages arriving on the transmission queue and transmit them as they arrive.

When you use triggering or run channels as threads, you will need to start the channel initiator to monitor the initiation queue. Use the `runmqchi` command for this.

However, TCP and LU 6.2 do provide other capabilities:

- For TCP on OS/2, Compaq OpenVMS Alpha, and UNIX systems, inetd (or an equivalent TCP/IP service on OpenVMS) can be configured to start a channel. This will be started as a separate process.
- For LU 6.2 in OS/2, using Communications Manager/2 it is possible to configure the Attach Manager to start a channel. This will be started as a separate process.
- For LU 6.2 in UNIX systems, configure your SNA product to start the LU 6.2 responder process.
- For LU 6.2 in Windows systems, using SNA Server you can use TpStart (a utility provided with SNA Server) to start a channel. This will be started as a separate process.
- For LU 6.2 in Compaq OpenVMS Alpha systems, use the `runmqlsr` command to start the LU 6.2 responder process.
- For LU 6.2 in Compaq NonStop Kernel, the `strmqm` command normally starts the LU 6.2 responder process if your channel is defined AUTOSTART (ENABLED). To start the process manually, use the `runmqsc` or `runmqcli` command.

Use of the Start option always causes the channel to re-synchronize, where necessary.

For the start to succeed:

- Channel definitions, local and remote, must exist. If there is no appropriate channel definition for a receiver or server-connection channel, a default one is created automatically if the channel is auto-defined. See "Channel auto-definition exit program" on page 583.
Channel functions

- Transmission queue must exist, and have no other channels using it.
- MCAs, local and remote, must exist.
- Communication link must be available.
- Queue managers must be running, local and remote.
- Message channel must not be already running.

A message is returned to the screen confirming that the request to start a channel has been accepted. For confirmation that the start command has succeeded, check the error log, or use DISPLAY CHSTATUS. The error logs are:

**OS/2 and Windows**

\qm\qmgs\qmname\errors\AMQERR01.LOG (for each queue manager called qmname)

\qm\qmgs\@SYSTEM\errors\AMQERR01.LOG (for general errors)

**Note:** On Windows systems, you still also get a message in the Windows systems application event log.

**Compaq OpenVMS Alpha**

MQS_ROOT:[MQM.QMGRS.QMNAME.ERRORS]AMQERR01.LOG (for each queue manager called qmname)

MQS_ROOT:[MQM.QMGRS.$SYSTEM.ERRORS]AMQERR01.LOG (for general errors)

**Compaq NonStop Kernel**

The location of the error logs depends on whether the queue manager name is known and whether the error is associated with a client.

- If the queue manager name is known and the queue manager is available:
  
  <QMVOL>.<SUBVOL>L.MQERRLG1

- If the queue manager is not available:
  
  <MQSVOL>.ZMQSSYS.MQERRLG1

**UNIX systems**

/var/mqm/qmgrs/qmname/errors/AMQERR01.LOG (for each queue manager called qmname)

/var/mqm/qmgrs/@SYSTEM/errors/AMQERR01.LOG (for general errors)

---

**Stop**

Use the MQSC command STOP CHANNEL to request the channel to stop activity. The channel will not start a new batch of messages until the operator starts the channel again. (For information about restarting stopped channels, see “Restarting stopped channels” on page 69).

This command can be issued to a channel of any type except MQCHT_CLNTCONN.

You can select the type of stop you require:

**Stop quiesce example**

STOP CHANNEL(QM1.TO.QM2) MODE(QUIESCE)
This command requests the channel to close down in an orderly way. The current batch of messages is completed and the syncpoint procedure is carried out with the other end of the channel.

**Note:** If the channel is idle this command will not terminate a receiving channel.

### Stop force example

```shell
STOP CHANNEL(QM1.TO.QM2) MODE(FORCE)
```

This option stops the channel immediately, but does not terminate the channel’s thread or process. The channel does not complete processing the current batch of messages, and can, therefore, leave the channel in doubt. In general, it is recommended that operators use the quiesce stop option.

### Stop terminate example

```shell
STOP CHANNEL(QM1.TO.QM2) MODE(TERMINATE)
```

This option stops the channel immediately, and terminates the channel’s thread or process.

### Reset

Use the MQSC command RESET CHANNEL to change the message sequence number. This command is available for any message channel, but not for MQI channels (client-connection or server-connection). The first message starts the new sequence the next time the channel is started.

If the command is issued on a sender or server channel, it informs the other side of the change when the channel is restarted.

### Resolve

Use the MQSC command RESOLVE CHANNEL when messages are held in-doubt by a sender or server, for example because one end of the link has terminated, and there is no prospect of it recovering. The RESOLVE CHANNEL command accepts one of two parameters: BACKOUT or COMMIT. Backout restores messages to the transmission queue, while Commit discards them.

The channel program does not try to establish a session with a partner. Instead, it determines the logical unit of work identifier (LUWID) which represents the in-doubt messages. It then issues, as requested, either:
- BACKOUT to restore the messages to the transmission queue; or
- COMMIT to delete the messages from the transmission queue.

For the resolution to succeed:
- The channel must be inactive
- The channel must be in doubt
- The channel type must be sender or server
- A local channel definition must exist
- The local queue manager must be running
Channel functions
Chapter 9. Preparing WebSphere MQ for distributed platforms

This chapter describes the WebSphere MQ preparations required before DQM can be used in OS/2, Windows NT, Compaq OpenVMS Alpha, Compaq NonStop Kernel, and UNIX systems. It includes “Transmission queues and triggering” and “Channel programs” on page 123.

Transmission queues and triggering

Before a channel (other than a requester channel) can be started, the transmission queue must be defined as described in this chapter, and must be included in the message channel definition.

In addition, where needed, the triggering arrangement must be prepared with the definition of the necessary processes and queues.

Creating a transmission queue

Define a local queue with the USAGE attribute set to XMITQ for each sending message channel. If you want to make use of a specific transmission queue in your remote queue definitions, create a remote queue as shown below.

To create a transmission queue, use the WebSphere MQ Commands (MQSC), as shown in the following examples:

**Create transmission queue example**

```
DEFINE QLOCAL(QM2) DESCR('Transmission queue to QM2') USAGE(XMITQ)
```

**Create remote queue example**

```
DEFINE QREMOTE(PAYROLL) DESCR('Remote queue for QM2') +
XMITQ(QM2) RNAME(PAYROLL) RQMNAME(QM2)
```

The recommended name for the transmission queue is the queue manager name on the remote system, as shown in the examples above.

Triggering channels

An overview of triggering is given in “Triggering channels” on page 20, while it is described in depth in the WebSphere MQ Application Programming Guide. This description provides you with information specific to WebSphere MQ for UNIX and Windows systems, and MQSeries for OS/2 Warp, Compaq OpenVMS Alpha, and Compaq NonStop Kernel.

You can create a process definition in WebSphere MQ, defining processes to be triggered. Use the MQSC command DEFINE PROCESS to create a process definition naming the process to be triggered when messages arrive on a transmission queue. The USERDATA attribute of the process definition should contain the name of the channel being served by the transmission queue.

Alternatively, for WebSphere MQ for AIX, HP-UX, Solaris, and Windows, and MQSeries for Compaq Tru64 UNIX, and OS/2 Warp, you can eliminate the need for a process definition by specifying the channel name in the TRIGDATA attribute of the transmission queue.
Transmission queues and triggering

If you do not specify a channel name, the channel initiator searches the channel definition files until it finds a channel that is associated with the named transmission queue.

**Example definitions for triggering**

Define the local queue (Q3), specifying that trigger messages are to be written to the default initiation queue SYSTEM.CHANNEL.INITQ, to trigger the application (process P1) that starts channel (QM3.TO.QM4):

```
DEFINE QLOCAL(QM4) TRIGGER INITQ(SYSTEM.CHANNEL.INITQ) PROCESS(P1) USAGE (XMITQ)
```

Define the application (process P1) to be started:
```
DEFINE PROCESS(P1) USERDATA(QM3.TO.QM4)
```

**Examples for WebSphere MQ for AIX, iSeries, HP-UX, Linux, Solaris, and Windows systems, and MQSeries V5.1 for Compaq Tru64 UNIX, and OS/2 Warp**

Define the local queue (Q3), specifying that trigger messages are to be written to the initiation queue (IQ) to trigger the application that starts channel (QM3.TO.QM4):

```
DEFINE QLOCAL(QM4) TRIGGER INITQ(SYSTEM.CHANNEL.INITQ) + USAGE (XMITQ) TRIGDATA(QM3.TO.QM4)
```

**Starting the channel initiator**

Triggering is implemented using the channel initiator process. On MQSeries for AT&T GIS UNIX, Compaq OpenVMS Alpha, SINIX and DC/OSx, and Compaq NonStop Kernel, this process is started with the run channel initiator command, `runmqchi`, or (on distributed platforms except Compaq NonStop Kernel) with the MQSC command START CHINIT. For example, to use the `runmqchi` command to start the default initiation queue for the default queue manager, enter:

```
rungqchi
```

Whichever command you use, specify the name of the initiation queue on the command, unless you are using the default initiation queue. For example, to use the `runmqchi` command to start queue IQ for the default queue manager, enter:

```
rungqchi -q IQ
```

To use the START CHINIT command (not on Compaq NonStop Kernel), enter:
```
START CHINIT INITQ(IQ)
```

**Note:** Compaq NonStop Kernel also allows control of the channel initiator from the PATHWAY environment. This is the recommended method. For more information about this, see the *MQSeries for Compaq NonStop Kernel System Administration* book.

In WebSphere MQ for AIX, iSeries, HP-UX, Linux, Solaris, and Windows systems, and MQSeries V5.1 for Compaq Tru64 UNIX, and OS/2 Warp, a channel initiator is started automatically and the number of channel initiators that you can start is limited. The default limit is 3. You can change this using MAXINITIATORS in the qm.ini file for AIX, HP-UX, OS/2 Warp, and Solaris, and in the registry for Windows systems.

See the *WebSphere MQ System Administration Guide* book for WebSphere MQ for AIX, iSeries, HP-UX, Linux, Solaris, and Windows systems, and MQSeries V5.1 for Compaq Tru64 UNIX, and OS/2 Warp for details of the run channel initiator command, and the other control commands.
Stopping the channel initiator

The default channel initiator is started automatically when you start a queue manager. Similarly, it is stopped automatically when a queue manager is stopped.

However, if you need to stop a channel initiator but not the queue manager, you should inhibit the queue that the initiator queue is reading from. To do this, you disable the GET attribute of the initiation queue. To restart the channel initiator, simply use the runmqchi command.

The consequences of stopping a channel initiator are:

- If you stop the only channel initiator running, no channels that you have attempted to start will be retried.
- If you have more than one channel initiator running, channels that have a transmission queue configured with this initiation queue are not automatically started. However, those channels configured for connection retries will continue to be retried.

Channel programs

There are different types of channel programs (MCAs) available for use at the channels. The names are shown in the following tables.

Table 10. Channel programs for OS/2 and Windows systems

<table>
<thead>
<tr>
<th>Program name</th>
<th>Direction of connection</th>
<th>Communication</th>
</tr>
</thead>
<tbody>
<tr>
<td>RUNMQLSR</td>
<td>Inbound</td>
<td>Any</td>
</tr>
<tr>
<td>ENDMQLSR</td>
<td></td>
<td></td>
</tr>
<tr>
<td>AMQCRS6A</td>
<td>Inbound</td>
<td>LU 6.2</td>
</tr>
<tr>
<td>AMQCRSTA</td>
<td>Inbound</td>
<td>TCP</td>
</tr>
<tr>
<td>RUNMQCHL</td>
<td>Outbound</td>
<td>Any</td>
</tr>
<tr>
<td>RUNMQCHI</td>
<td>Outbound</td>
<td></td>
</tr>
</tbody>
</table>

Table 11. Channel programs for UNIX systems, Compaq OpenVMS Alpha, and Compaq NonStop Kernel

<table>
<thead>
<tr>
<th>Program name</th>
<th>Direction of connection</th>
<th>Communication</th>
</tr>
</thead>
<tbody>
<tr>
<td>amqcrs6a</td>
<td>Inbound</td>
<td>LU 6.2</td>
</tr>
<tr>
<td>amqcrsta</td>
<td>Inbound</td>
<td>TCP and DECnet for Compaq OpenVMS Alpha</td>
</tr>
<tr>
<td>runmqchl</td>
<td>Outbound</td>
<td>TCP for UNIX systems</td>
</tr>
<tr>
<td>runmqlsr</td>
<td>Inbound</td>
<td>LU 6.2 for Compaq OpenVMS Alpha and Compaq NonStop Kernel and TCP for UNIX systems</td>
</tr>
<tr>
<td>runmqchi</td>
<td>Outbound</td>
<td>Any</td>
</tr>
</tbody>
</table>

RUNMQLSR (Run WebSphere MQ listener), ENDMQLSR (End WebSphere MQ listener), and RUNMQCHL (Run WebSphere MQ channel) are control commands that you can enter at the command line. AMQCRS6A and AMQCRSTA are programs that, if you are using SNA, you define as transaction programs, or, if you...
Channel programs

are using TCP, you define in the INETD.LST file for OS/2 or Windows systems or the inetd.conf file for UNIX systems. Examples of the use of these channel programs are given in the following chapters.

Other things to consider

Here are some other topics that you should consider when preparing WebSphere MQ for distributed queue management.

Undelivered-message queue

A DLQ handler is provided with WebSphere MQ for z/OS, MQSeries for OS/2 Warp, WebSphere MQ on UNIX systems, Compaq OpenVMS Alpha and Compaq NonStop Kernel. See the WebSphere MQ System Administration Guide book for WebSphere MQ for AIX, iSeries, HP-UX, Linux, Solaris, and Windows systems, and MQSeries V5.1 for Compaq Tru64 UNIX, and OS/2 Warp for information about this.

Queues in use

MCAs for receiver channels may keep the destination queues open even when messages are not being transmitted; this results in the queues appearing to be “in use”.

Security of WebSphere MQ objects

This section deals with remote messaging aspects of security.

You need to provide users with authority to make use of the WebSphere MQ facilities, and this is organized according to actions to be taken with respect to objects and definitions. For example:

• Queue managers can be started and stopped by authorized users
• Applications need to connect to the queue manager, and have authority to make use of queues
• Message channels need to be created and controlled by authorized users
• Objects are kept in libraries, and access to these libraries may be restricted

The message channel agent at a remote site needs to check that the message being delivered originated from a user with authority to do so at this remote site. In addition, as MCAs can be started remotely, it may be necessary to verify that the remote processes trying to start your MCAs are authorized to do so. There are three possible ways for you to deal with this:

1. Specify PUTAUT=CTX in the channel definition to indicate that messages must contain acceptable context authority, otherwise they will be discarded.
2. Implement user exit security checking to ensure that the corresponding message channel is authorized. The security of the installation hosting the corresponding channel ensures that all users are properly authorized, so that you do not need to check individual messages.
3. Implement user exit message processing to ensure that individual messages are vetted for authorization.

On UNIX systems, Compaq OpenVMS Alpha, and Compaq NonStop Kernel

Administration users must be part of the mqm group on your system (including root) if this ID is going to use WebSphere MQ administration commands. In Compaq OpenVMS Alpha, the user must hold the mqm identifier.
You should always run amqcrsta as the “mqm” user ID.

User IDs on UNIX systems, Compaq OpenVMS Alpha: In Compaq OpenVMS Alpha, all user IDs are displayed in uppercase. The queue manager converts all uppercase or mixed case user identifiers into lowercase, before inserting them into the context part of a message, or checking their authorization. All authorizations should therefore be based only on lowercase identifiers.

Message descriptor extension (MQMDE): When the listener program (amqcrsta, for example) is started by INETD it inherits the locale from INETD. It is possible that the MQMDE will not be honored and will be placed on the queue as message data.

To ensure that the MQMDE is honored (merged) the locale must be set correctly. The locale set by INETD may not match that chosen for other locales used by WebSphere MQ processes.

To set the locale, create a shell script which sets the locale environment variables LANG, LC_COLLATE, LC_CTYPE, LC_MONETARY, LC_NUMERIC, LC_TIME, and LC_MESSAGES to the locale used for other WebSphere MQ processes. In the same shell script call the listener program. Modify the inetd.conf file to call your shell script in place of the listener program.

On Windows systems
Administration users must be part of both the mqm group and the administrators group on Windows systems if this ID is going to use WebSphere MQ administration commands.

User IDs on Windows systems: On Windows systems, if there is no message exit installed, the queue manager converts any uppercase or mixed case user identifiers into lowercase, before inserting them into the context part of a message, or checking their authorization. All authorizations should therefore be based only on lowercase identifiers.

User IDs across systems
Platforms other than Windows systems and UNIX systems use uppercase characters for user IDs. To allow Windows systems and UNIX systems to use lowercase user IDs, the following conversions are carried out by the message channel agent (MCA) on these platforms:

At the sending end
The alpha characters in all user IDs are converted to uppercase, if there is no message exit installed.

At the receiving end
The alpha characters in all user IDs are converted to lowercase, if there is no message exit installed.

Note that the automatic conversions are not carried out if you provide a message exit on UNIX systems and Windows systems for any other reason.

User IDs on OS/2
The user identifier service enables queue managers running under OS/2 to obtain a user-defined user ID. This is described in the WebSphere MQ Programmable Command Formats and Administration Interface book.
Other things to consider

System extensions and user-exit programs

A facility is provided in the channel definition to allow extra programs to be run at defined times during the processing of messages. These programs are not supplied with WebSphere MQ, but may be provided by each installation according to local requirements.

In order to run, these user-exit programs must have predefined names and be available on call to the channel programs. The names of the user-exit programs are included in the message channel definitions.

There is a defined control block interface for handing over control to these programs, and for handling the return of control from these programs.

The precise places where these programs are called, and details of control blocks and names, are to be found in Part 7, “Further intercommunication considerations” on page 567.

Running channels and listeners as trusted applications

If performance is an important consideration in your environment and your environment is stable, you can choose to run your channels and listeners as trusted, that is, using the fastpath binding. There are two factors that influence whether or not channels and listeners run as trusted.

- The environment variable MQ_CONNECT_TYPE=FASTPATH or MQ_CONNECT_TYPE=STANDARD. This is case sensitive. If you specify a value that is not valid it is ignored.
- MQIBindType in the Channels stanza of the qm.ini or registry file. You can set this to FASTPATH or STANDARD and it is not case sensitive. The default is STANDARD.

You can use MQIBindType in association with the environment variable to achieve the desired affect as follows:

<table>
<thead>
<tr>
<th>MQIBindType</th>
<th>Environment variable</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>STANDARD</td>
<td>UNDEFINED</td>
<td>STANDARD</td>
</tr>
<tr>
<td>FASTPATH</td>
<td>UNDEFINED</td>
<td>FASTPATH</td>
</tr>
<tr>
<td>STANDARD</td>
<td>STANDARD</td>
<td>STANDARD</td>
</tr>
<tr>
<td>FASTPATH</td>
<td>STANDARD</td>
<td>STANDARD</td>
</tr>
<tr>
<td>STANDARD</td>
<td>FASTPATH</td>
<td>STANDARD</td>
</tr>
<tr>
<td>FASTPATH</td>
<td>FASTPATH</td>
<td>FASTPATH</td>
</tr>
</tbody>
</table>

In summary, there are only two ways of actually making channels and listeners run as trusted:

1. By specifying MQIBindType=FASTPATH in qm.ini or registry and not specifying the environment variable.
2. By specifying MQIBindType=FASTPATH in qm.ini or registry and setting the environment variable to FASTPATH.

You are recommended to run channels and listeners as trusted only in a stable environment in which you are not, for example, testing applications or user exits that may abend or need to be cancelled. An errant application could compromise
the integrity of your queue manager. However, if your environment is stable and if performance is an important issue, you may choose to run channels and listeners as trusted.

**Note:** If you are using MQSeries for Compaq OpenVMS Alpha, the option on the MQ_CONNECT_TYPE is FAST, not FASTPATH.

### What next?

When you have made the preparations described in this chapter you are ready to set up communications. Proceed to one of the following chapters, depending on what platform you are using:

- Chapter 10, “Setting up communication for OS/2 and Windows” on page 129
- Chapter 13, “Setting up communication in UNIX systems” on page 191
- Chapter 19, “Setting up communication in Compaq OpenVMS Alpha systems” on page 295
- Chapter 20, “Setting up communication in Compaq NonStop Kernel” on page 307
What next
Chapter 10. Setting up communication for OS/2 and Windows

DQM is a remote queuing facility for WebSphere MQ. It provides channel control programs for the queue manager which form the interface to communication links, controllable by the system operator. The channel definitions held by distributed-queuing management use these connections.

When a distributed-queuing management channel is started, it tries to use the connection specified in the channel definition. For this to succeed, it is necessary for the connection to be defined and available. This chapter explains how to do this. You may also find it helpful to refer to Chapter 11, “Example configuration - MQSeries for OS/2 Warp” on page 143 or Chapter 12, “Example configuration - IBM WebSphere MQ for Windows” on page 167.

For UNIX systems see Chapter 13, “Setting up communication in UNIX systems” on page 191. For Compaq OpenVMS Alpha, see Chapter 19, “Setting up communication in Compaq OpenVMS Alpha systems” on page 295.

Deciding on a connection

There are four forms of communication for MQSeries for OS/2 Warp and Windows systems:

- TCP
- LU 6.2
- NetBIOS
- SPX

Each channel definition must specify only one protocol as the Transmission protocol (Transport Type) attribute. One or more protocols may be used by a queue manager.

For WebSphere MQ clients, it may be useful to have alternative channels using different transmission protocols. See the WebSphere MQ Clients book.
Defining a TCP connection

The channel definition at the sending end specifies the address of the target. A listener program must be run at the receiving end.

Sending end

Specify the host name, or the TCP address of the target machine, in the Connection name field of the channel definition. The port to connect to will default to 1414. Port number 1414 is assigned by the Internet Assigned Numbers Authority to WebSphere MQ.

To use a port number other than the default, change the connection name field thus:

Connection Name OS2ROG3(1822)

where 1822 is the port required. (This must be the port that the listener at the receiving end is listening on.)

You can change the default port number by specifying it in the queue manager configuration file (qm.ini) for MQSeries for OS/2 Warp and the registry for WebSphere MQ for Windows:

TCP:
  Port=1822

Note: To select which TCP/IP port number to use, WebSphere MQ uses the first port number it finds in the following sequence:

1. The port number explicitly specified in the channel definition or command line. This number allows the default port number to be overridden for a channel.
2. The port attribute specified in the TCP stanza in the qm.ini file. This number allows the default port number to be overridden for a queue manager.
3. The value specified for 'MQSeries' in the '/etc/services' file. This number allows the default port number to be overridden for a machine.
4. The default value of 1414. This is the number assigned to WebSphere MQ by the Internet Assigned Numbers Authority.

For more information about the values you set using qm.ini, see Appendix D, "Configuration file stanzas for distributed queuing" on page 715.

Receiving on TCP

Receiving channel programs are started in response to a startup request from the sending channel. To do this, a listener program has to be started to detect incoming network requests and start the associated channel.

You should use either the TCP/IP listener (INETD) (not for Windows) or the WebSphere MQ listener.

Using the TCP/IP listener

To use INETD to start channels on OS/2, two files must be configured:

1. Add a line in the TCPIP\ETC\SERVICES file:

MQSeries 1414/tcp
Defining a TCP connection

where 1414 is the port number required for WebSphere MQ. You can change this but it must match the port number specified at the sending end.

2. Add a line to the TCPIP\ETC\INETD.LST file:

MQSeries tcp C:\MQM\BIN\AMQCRSTA [-m QMName]

The last part in square brackets is optional and is not required for the default queue manager. If your MQSeries for OS/2 Warp is installed on a different drive, replace the C: above with the correct drive letter.

It is possible to have more than one queue manager on the machine. You must add a line to each of the two files, as above, for each of the queue managers. For example:

MQSeries2 1822/tcp

Now stop, and then start the inetd program, before continuing.

Using the TCP listener backlog option

When receiving on TCP, a maximum number of outstanding connection requests is set. This can be considered a backlog of requests waiting on the TCP port for the listener to accept the request. The default listener backlog values are shown in Table 12.

Table 12. Default outstanding connection requests on OS/2 and Windows

<table>
<thead>
<tr>
<th>Platform</th>
<th>Default listener backlog value</th>
</tr>
</thead>
<tbody>
<tr>
<td>OS/2 Warp</td>
<td>10</td>
</tr>
<tr>
<td>Windows Server</td>
<td>100</td>
</tr>
<tr>
<td>Windows Workstation</td>
<td>100</td>
</tr>
</tbody>
</table>

If the backlog reaches the values shown in Table 12 the TCP/IP connection is rejected and the channel will not be able to start.

For MCA channels, this results in the channel going into a RETRY state and retrying the connection at a later time.

For client connections, the client receives an MQRC_Q_MGR_NOT_AVAILABLE reason code from MQCONN and should retry the connection at a later time.

However, to avoid this error, you can add an entry in the qm.ini file or in the registry for Windows:

TCP:
ListenerBacklog = n

This overrides the default maximum number of outstanding requests (see Table 12) for the TCP/IP listener.

Note: Some operating systems support a larger value than the default. If necessary, this can be used to avoid reaching the connection limit.

To run the listener with the backlog option switched on, use the RUNMQLSR -b command. For information about the RUNMQLSR command, see the WebSphere MQ System Administration Guide book.
Defining a TCP connection

**Using the WebSphere MQ listener**
To run the Listener supplied with WebSphere MQ, that starts new channels as threads, use the RUNMQLSR command. For example:

```
RUNMQLSR -t tcp [-m QMNAME] [-p 1822]
```

The square brackets indicate optional parameters; QMNAME is not required for the default queue manager, and the port number is not required if you are using the default (1414).

For the best performance, run the WebSphere MQ listener as a trusted application as described in “Running channels and listeners as trusted applications” on page 126. See the [WebSphere MQ Application Programming Guide](#) for information about trusted applications.

You can stop all WebSphere MQ listeners running on a queue manager that is inactive, using the command:

```
ENDMQLSR [-m QMNAME]
```

If you do not specify a queue manager name, the default queue manager is assumed.

**Using the TCP/IP SO_KEEPALIVE option**
If you want to use the SO_KEEPALIVE option (as discussed in “Checking that the other end of the channel is still available” on page 66) you need to add the following entry to your queue manager configuration file (qm.ini):

```
TCP:
    KeepAlive=yes
```

If you are using OS/2, you must then issue the following command:

```
inetcfg keepalive=value
```

where `value` is the time interval in minutes.

On Windows, the TCP configuration registry value for KeepAliveTime controls the interval that elapses before the connection will be checked. The default is two hours. For information about changing this value, see the Microsoft article [TCP/IP and NBT Configuration Parameters for Windows NT 3.5](#) (PSS ID number Q120642).

### Defining an LU 6.2 connection

SNA must be configured so that an LU 6.2 conversation can be established between the two machines. Then proceed as follows.

See the [Multiplatform APPC Configuration Guide](#) for OS/2 examples, and the following table for information.

**Table 13. Settings on the local OS/2 or Windows system for a remote queue manager platform**

<table>
<thead>
<tr>
<th>Remote platform</th>
<th>TPNAME</th>
<th>TPATH</th>
</tr>
</thead>
<tbody>
<tr>
<td>z/OS or OS/390 or MVS/ESA without CICS</td>
<td>The same as in the corresponding side information on the remote queue manager.</td>
<td>-</td>
</tr>
<tr>
<td>z/OS or OS/390 or MVS/ESA using CICS</td>
<td>CKRC (sender) CKSV (requester) CKRC (server)</td>
<td>-</td>
</tr>
</tbody>
</table>
Table 13. Settings on the local OS/2 or Windows system for a remote queue manager platform (continued)

<table>
<thead>
<tr>
<th>Remote platform</th>
<th>TPNAME</th>
<th>TPPATH</th>
</tr>
</thead>
<tbody>
<tr>
<td>OS/400</td>
<td>The same as the compare value in the routing entry on the OS/400 system.</td>
<td>-</td>
</tr>
<tr>
<td>OS/2</td>
<td>As specified in the OS/2 Run Listener command, or defaulted from the OS/2 queue manager configuration file.</td>
<td>&lt;drive&gt;:\mqm\bin\amqcrs6a</td>
</tr>
<tr>
<td>UNIX systems</td>
<td>The same as in the corresponding side information on the remote queue manager.</td>
<td>mqmtop/bin/amqcrs6a</td>
</tr>
<tr>
<td>Windows</td>
<td>As specified in the Windows Run Listener command, or the invokable Transaction Program that was defined using TpSetup on Windows.</td>
<td>&lt;drive&gt;:\mqm\bin\amqcrs6a</td>
</tr>
</tbody>
</table>

If you have more than one queue manager on the same machine, ensure that the TPnames in the channel definitions are unique.

### Sending end for OS/2

Establish a valid session between the two machines. The local LU that WebSphere MQ uses is decided in the following order:

1. Specify the LU that will be used. In the queue manager configuration file (qm.ini), under the LU 6.2 section add the line:
   ```
   LOCALLU = Your_LU_Name
   ```
   For more information about the values you set using qm.ini, see Appendix D, "Configuration file stanzas for distributed queuing" on page 715.

2. Specify the environment variable:
   ```
   APPNLLU = Your_LU_Name
   ```

3. If this has not been specified, your default LU will be used.

When you define a WebSphere MQ channel that will use the LU 6.2 connection, the Connection name (CONNAME) channel attribute specifies the fully-qualified name of the partner LU. as defined in the local Communications Manager/2 profile.

SECURITY PROGRAM is always used when WebSphere MQ attempts to establish an SNA session.

### Sending end for Windows

Create a CPI-C side object (symbolic destination) from the administration application of the LU 6.2 product you are using, and enter this name in the Connection name field in the channel definition. Also create an LU 6.2 link to the partner.

In the CPI-C side object enter the partner LU Name at the receiving machine, the TP Name and the Mode Name. For example:
Receiving on LU 6.2

Receiving channel programs are started in response to a startup request from the sending channel. To do this, a listener program has to be started to detect incoming network requests and start the associated channel. You start this listener program with the RUNMQLSR command, giving the TpName to listen on. Alternatively, you can use Attach Manager in Communications Manager/2 for OS/2, or TpStart under SNA Server for Windows.

Using the RUNMQLSR command

Example of the command to start the listener:

RUNMQLSR -t LU62 -n RECV [-m QMNAME]

where RECV is the TpName that is specified at the other (sending) end as the “TpName to start on the remote side”. The last part in square brackets is optional and is not required for the default queue manager.

It is possible to have more than one queue manager running on one machine. You must assign a different TpName to each queue manager, and then start a listener program for each one. For example:

RUNMQLSR -t LU62 -m QM1 -n TpName1
RUNMQLSR -t LU62 -m QM2 -n TpName2

For the best performance, run the WebSphere MQ listener as a trusted application as described in “Running channels and listeners as trusted applications” on page 126. See the WebSphere MQ Application Programming Guide for information about trusted applications.

You can stop all WebSphere MQ listeners running on a queue manager that is inactive, using the command:

ENDMQLSR [-m QMNAME]

If you do not specify a queue manager name, the default queue manager is assumed.

Using Communications Manager/2 on OS/2

If you are going to use Attach Manager in Communications Manager/2 to start the listener program, you must specify the Program parameter string or parm_string in addition to the TPNAMESPACE and TPPATH.

You can do this using the panel configuration in Communications Manager/2 or, alternatively, you can edit your NDF file directly (see the heading “Define Transaction Programs” in the Multiplatform APPC Configuration Guide).

Panel configuration:

These are the entries required on the TP definition panel:

Transaction Program (TP) name : AMQCRS6A
OS/2 program path and file name: c:\mqm\bin\amqcrs6a.exe
Program parameter string : -n AMQCRS6A

NDF file configuration: Your node definitions file (.ndf) must contain a define_tp command. The following example shows what must be included:
Using Microsoft SNA Server on Windows
You can use TpSetup (from the SNA Server SDK) to define an invokable TP that then drives amqcrs6a.exe, or you can set various registry values manually. The parameters that should be passed to amqcrs6a.exe are:

```
-m QM -n TpName
```

where QM is the Queue Manager name and TpName is the TP Name. See the Microsoft SNA Server APPC Programmers Guide or the Microsoft SNA Server CPI-C Programmers Guide for more information.

Defining a NetBIOS connection
WebSphere MQ uses three types of NetBIOS resource when establishing a NetBIOS connection to another WebSphere MQ product: sessions, commands, and names. Each of these resources has a limit, which is established either by default or by choice during the installation of NetBIOS.

Each running channel, regardless of type, uses one NetBIOS session and one NetBIOS command. The IBM NetBIOS implementation allows multiple processes to use the same local NetBIOS name. Therefore, only one NetBIOS name needs to be available for use by WebSphere MQ. Other vendors’ implementations, for example Novell’s NetBIOS emulation, require a different local name per process. Verify your requirements from the documentation for the NetBIOS product you are using.

In all cases, ensure that sufficient resources of each type are already available, or increase the maximums specified in the configuration. Any changes to the values will require a system restart.

During system startup, the NetBIOS device driver displays the number of sessions, commands, and names available for use by applications. These resources are available to any NetBIOS-based application that is running on the same system. Therefore, it is possible for other applications to consume these resources before WebSphere MQ needs to acquire them. Your LAN network administrator should be able to clarify this for you.

Defining the WebSphere MQ local NetBIOS name
The local NetBIOS name used by WebSphere MQ channel processes can be specified in three ways. In order of precedence they are:

1. The value specified in the -l parameter of the RUNMQLSR command, for example:
   
   ```
   RUNMQLSR -t NETBIOS -l my_station
   ```

2. The MQNAME environment variable whose value is established by the command:
   
   ```
   SET MQNAME=my_station
   ```

You can set the MQNAME value for each process. Alternatively, you may set it at a system level — in the CONFIG.SYS file on OS/2 or in the Windows registry.
Defining a NetBIOS connection

If you are using a NetBIOS implementation that requires unique names, you must issue a SET™ MQNAME command in each window in which a WebSphere MQ process is started. The MQNAME value is arbitrary but it must be unique for each process.

3. The NETBIOS stanza in the queue manager configuration file qm.ini or in the Windows registry. For example:
   
   ```
   NETBIOS:
   
   LocalName=my_station
   ```

   **Notes:**
   1. Due to the variations in implementation of the NetBIOS products supported, you are advised to make each NetBIOS name unique in the network. If you do not, unpredictable results may occur. If you have problems establishing a NetBIOS channel and there are error messages in the queue-manager error log showing a NetBIOS return code of X'15', review your use of NetBIOS names.
   2. On Windows you cannot use your machine name as the NetBIOS name because Windows already uses it.
   3. Sender channel initiation requires that a NetBIOS name be specified either via the MQNAME environment variable or the LocalName in the qm.ini file or in the Windows registry.

Establishing the queue manager NetBIOS session, command, and name limits

The queue manager limits for NetBIOS sessions, commands, and names can be specified in two ways. In order of precedence they are:

1. The values specified in the RUNMQLSR command:
   
   ```
   -s Sessions
   -e Names
   -o Commands
   ```
   
   If the -m operand is not specified in the command, the values will apply only to the default queue manager.

2. The NETBIOS stanza in the queue manager configuration file qm.ini or in the Windows registry. For example:
   
   ```
   NETBIOS:
   
   NumSess=Qmgr_max_sess
   NumCmds=Qmgr_max_cmds
   NumNames=Qmgr_max_names
   ```

Establishing the LAN adapter number

For channels to work successfully across NetBIOS, the adapter support at each end must be compatible. WebSphere MQ allows you to control the choice of adapter number (lana) by using the AdapterNum value in the NETBIOS stanza of your qm.ini file or the Windows registry and by specifying the -a parameter on the runmqlsr command.

The default LAN adapter number used by WebSphere MQ for NetBIOS connections is 0. Verify the adapter number being used on your system as follows:

On OS/2 the adapter number used by NetBIOS on your system can be viewed in the PROTOCOL.INI file or the LANTRAN.LOG file found in the \IBMCOM directory.
Defining a NetBIOS connection

On Windows, view the information displayed in the NetBIOS Interface pop-up window. This is accessible by selecting the Network option, which is one of many options displayed when opening the Control icon from the Main Window. Windows can assign multiple ‘logical’ adapter numbers to one physical LAN adapter. The installation default for ‘logical’ adapter number 0 is NetBIOS running over a TCP network, not a Token-Ring network. This is not necessary for WebSphere MQ. You should select logical adapter number 1, which is native NetBIOS. WebSphere MQ for Windows uses the ‘logical’ adapter number for communication.

Specify the correct value in the NETBIOS stanza of the queue manager configuration file, qm.ini, or the Windows registry:

```
NETBIOS:
    AdapterNum=n
```

where n is the correct LAN adapter number for this system.

Initiating the connection

To initiate the connection, follow these steps at the sending end:

1. Define the NetBIOS station name using the MQNAME or LocalName value as described above.
2. Verify the LAN adapter number being used on your system and specify the correct file using the AdapterNum as described above.
3. In the ConnectionName field of the channel definition, specify the NetBIOS name being used by the target listener program. On Windows, NetBIOS channels must be run as threads. Do this by specifying MCATYPE(THREAD) in the channel definition.

```
DEFINE CHANNEL (chname) CHLTYPE(SDR) +
    TRPTYPE(NETBIOS) +
    CONNAME(your_station) +
    XMITQ(xmitq) +
    MCATYPE(THREAD) +
    REPLACE
```

Target listener

At the receiving end, follow these steps:

1. Define the NetBIOS station name using the MQNAME or LocalName value as described above.
2. Verify the LAN adapter number being used on your system and specify the correct file using the AdapterNum as described above.
3. Define the receiver channel:

```
DEFINE CHANNEL (chname) CHLTYPE(RCVR) +
    TRPTYPE(NETBIOS) +
    REPLACE
```

4. Start the WebSphere MQ listener program to establish the station and make it possible to contact it. For example:

```
RUNMQSLR -t NETBIOS -l your_station [-m qmgr]
```

This command establishes your_station as a NetBIOS station waiting to be contacted. The NetBIOS station name must be unique throughout your NetBIOS network.
Defining a NetBIOS connection

For the best performance, run the WebSphere MQ listener as a trusted application as described in “Running channels and listeners as trusted applications” on page 126. See the WebSphere MQ Application Programming Guide for information about trusted applications.

You can stop all WebSphere MQ listeners running on a queue manager that is inactive, using the command:

ENDMQLSR [-m QMNAME]

If you do not specify a queue manager name, the default queue manager is assumed.

Defining an SPX connection

The channel definition at the sending end specifies the address of the target. A listener program must be run at the receiving end.

Sending end

If the target machine is remote, specify the SPX address of the target machine in the Connection name field of the channel definition.

The SPX address is specified in the following form:

\textit{network.node(socket)}

where:

- \textit{network} is the 4-byte network address of the network on which the remote machine resides,
- \textit{node} is the 6-byte node address, which is the LAN address of the LAN adapter in the remote machine,
- \textit{socket} is the 2-byte socket number on which the remote machine will listen.

If the local and remote machines are on the same network then the network address need not be specified. If the remote end is listening on the default socket (5E86) then the socket need not be specified.

An example of a fully specified SPX address specified in the CONNAME parameter of an MQSC command is:

\texttt{CONNAME('00000001.08005A7161E5(5E87)')}

In the default case, where the machines are both on the same network, this becomes:

\texttt{CONNAME(08005A7161E5)}

The default socket number may be changed by specifying it in the queue manager configuration file (qm.ini) or the Windows registry:

\texttt{SPX:
Socket=5E87}

For more information about the values you set using qm.ini or the Windows registry, see Appendix D, “Configuration file stanzas for distributed queuing” on page 715.
Using the SPX KEEPALIVE option (OS/2 only)
If you want to use the KEEPALIVE option (as discussed in “Checking that the other end of the channel is still available” on page 66) you need to add the following entry to your queue manager configuration file (qm.ini):

```
SPX:
  KeepAlive=yes
```

You can use the timeouts described in “IPX/SPX parameters” on page 140 to adjust the behavior of KEEPALIVE.

Receiving on SPX
Receiving channel programs are started in response to a startup request from the sending channel. To do this, a listener program has to be started to detect incoming network requests and start the associated channel.

You should use the WebSphere MQ listener.

Using the SPX listener backlog option
When receiving on SPX, a maximum number of outstanding connection requests is set. This can be considered a backlog of requests waiting on the SPX port for the listener to accept the request. The default listener backlog values are shown in Table 14.

<table>
<thead>
<tr>
<th>Platform</th>
<th>Default listener backlog value</th>
</tr>
</thead>
<tbody>
<tr>
<td>OS/2 Warp</td>
<td>5</td>
</tr>
<tr>
<td>Windows Server</td>
<td>5</td>
</tr>
<tr>
<td>Windows Workstation</td>
<td>5</td>
</tr>
</tbody>
</table>

If the backlog reaches the values in Table 14, the reason code, MQRC_Q_MGR_NOT_AVAILABLE is received when trying to connect to the queue manager using MQCONN or MQCONNX. If this happens, it is possible to try to connect again.

However, to avoid this error, you can add an entry in the qm.ini file or in the registry for Windows:
```
SPX:
  ListenerBacklog = n
```

This overrides the default maximum number of outstanding requests (see Table 14) for the SPX listener.

Note: Some operating systems support a larger value than the default. If necessary, this can be used to avoid reaching the connection limit.

To run the listener with the backlog option switched on, use the RUNMQLSR -B command. For information about the RUNMQLSR command, see the WebSphere MQ System Administration Guide book.

Using the WebSphere MQ listener
To run the Listener supplied with WebSphere MQ, that starts new channels as threads, use the RUNMQLSR command. For example:

```
RUNMQLSR -t spx [-m QMNAME] [-x 5E87]
```
Defining an SPX connection

The square brackets indicate optional parameters; QMNAME is not required for the default queue manager, and the socket number is not required if you are using the default (5E86).

For the best performance, run the WebSphere MQ listener as a trusted application as described in “Running channels and listeners as trusted applications” on page 126. See the WebSphere MQ Application Programming Guide for information about trusted applications.

You can stop all WebSphere MQ listeners running on a queue manager that is inactive, using the command:

ENDMQLSR [-m QMNAME]

If you do not specify a queue manager name, the default queue manager is assumed.

IPX/SPX parameters

In most cases the default settings for the IPX/SPX parameters will suit your needs. However, you may need to modify some of them in your environment to tune its use for WebSphere MQ. The actual parameters and the method of changing them varies according to the platform and provider of SPX communications support. The following sections describe some of these parameters, particularly those that may influence the operation of WebSphere MQ channels and client connections.

OS/2

Please refer to the Novell Client for OS/2 documentation for full details of the use and setting of NET.CFG parameters.

The following IPX/SPX parameters can be added to the Novell NET.CFG file, and can affect WebSphere MQ SPX channels and client connections.

IPX:

sockets (range = 9 - 128, default 64)
This specifies the total number of IPX sockets available. WebSphere MQ channels use this resource, so depending on the number of channels and the requirements of other IPX/SPX applications, you may need to increase this value.

SPX:

sessions (default 16)
This specifies the total number of simultaneous SPX connections. Each WebSphere MQ channel or client connection uses one session. You may need to increase this value depending on the number of WebSphere MQ channels or client connections you need to run.

retry count (default = 12)
This controls the number of times an SPX session will resend unacknowledged packets. WebSphere MQ does not override this value.

verify timeout, listen timeout, and abort timeout (milliseconds)
These timeouts adjust the ‘Keepalive’ behavior. If an SPX sending end does not receive anything within the ‘verify timeout’ period, it sends a packet to the receiving end. It then waits for the duration of the ‘listen timeout’ for a response. If it still does not receive a response, it sends another packet and expects a response within the ‘abort timeout’ period.
DOS and Windows 3.1 client
Please refer to the Novell Client for DOS and MS Windows documentation for full details of the use and setting of NET.CFG parameters.

The following IPX/SPX parameters can be added to the Novell NET.CFG file, and can affect WebSphere MQ SPX channels and client connections.

IPX:

sockets (default = 20)
This specifies the total number of IPX sockets available. WebSphere MQ channels use this resource, so depending on the number of channels and the requirements of other IPX/SPX applications, you may need to increase this value.

retry count
This controls the number of times unacknowledged packets will be resent. WebSphere MQ does not override this value.

SPX:

connections (default 15)
This specifies the total number of simultaneous SPX connections. Each WebSphere MQ channel or client connection uses one session. You may need to increase this value depending on the number of WebSphere MQ channels or client connections you need to run.

Windows systems
Please refer to the Microsoft documentation for full details of the use and setting of the NWLink IPX and SPX parameters. The IPX/SPX parameters are in the following paths in the registry:

HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Service\NWLinkSPX\Parameters
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Service\NWLinkIPX\Parameters

Windows 95 and Windows 98
Please refer to the Microsoft documentation for full details of the use and setting of the IPX and SPX parameters. You access them by selecting Network option in the control panel, then double-clicking on IPX/SPX Compatible Transport.
DQM in distributed platforms
Chapter 11. Example configuration - MQSeries for OS/2 Warp

This chapter gives an example of how to set up communication links from MQSeries for OS/2 Warp to WebSphere MQ and MQSeries products on the following platforms:
- Windows systems
- AIX
- Compaq Tru64 UNIX
- HP-UX
- AT&T GIS UNIX
- Solaris
- OS/400
- z/OS without CICS
- VSE/ESA

First it describes the parameters needed for an LU 6.2 connection, then it guides you through the following tasks:
- "Establishing an LU 6.2 connection" on page 148
- "Establishing a TCP connection" on page 155
- "Establishing a NetBIOS connection" on page 157
- "Establishing an SPX connection" on page 157

Once the connection is established, you need to define some channels to complete the configuration. This is described in "MQSeries for OS/2 Warp configuration" on page 159.

See Chapter 7, “Example configuration chapters in this book” on page 101 for background information about this chapter and how to use it.

Configuration parameters for an LU 6.2 connection

Table 15 on page 144 presents a worksheet listing all the parameters needed to set up communication from OS/2 to one of the other WebSphere MQ platforms. The worksheet shows examples of the parameters, which have been tested in a working environment, and leaves space for you to fill in your own values. An explanation of the parameter names follows the worksheet. Use the worksheet in this chapter in conjunction with the worksheet in the chapter for the platform to which you are connecting.

This chapter shows how to use the values on the worksheet for:
- "Defining local node characteristics" on page 148
- "Connecting to a peer system" on page 151
- "Connecting to a host system" on page 153
- "Verifying the configuration" on page 154

Configuration worksheet

Use the following worksheet to record the values you will use for this configuration. Where numbers appear in the Reference column they indicate that the value must match that in the appropriate worksheet elsewhere in this book.

1. This platform has become NCR UNIX SVR4 MP-RAS, R3.0

© Copyright IBM Corp. 1993, 2002
### Table 15. Configuration worksheet for Communications Manager/2

<table>
<thead>
<tr>
<th>ID</th>
<th>Parameter Name</th>
<th>Reference</th>
<th>Example Used</th>
<th>User Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Configuration name</td>
<td></td>
<td>EXAMPLE</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Network ID</td>
<td></td>
<td>NETID</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Local node name</td>
<td></td>
<td>OS2PU</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Local node ID (hex)</td>
<td></td>
<td>05D 12345</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Local node alias name</td>
<td></td>
<td>OS2PU</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>LU name (local)</td>
<td></td>
<td>OS2LU</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>Alias (for local LU name)</td>
<td></td>
<td>OS2QMGR</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>Local transaction program (TP) name</td>
<td></td>
<td>MQSERIES</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>OS/2 program path and file name</td>
<td></td>
<td>c:\mqm\bin\amqcrs6a.exe</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>LAN adapter address</td>
<td></td>
<td>10005AFC5D83</td>
<td></td>
</tr>
</tbody>
</table>

#### Definition for local node

The examples that follow in this chapter refer back to the values in the ID column of this table. The entries in the Parameter Name column are explained in “Explaination of terms” on page 146.

### Connection to a Windows system

The values in this section of the table must match those used in Table 17 on page 168, as indicated.

<table>
<thead>
<tr>
<th>ID</th>
<th>Parameter Name</th>
<th>Reference</th>
<th>Example Used</th>
<th>User Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>11</td>
<td>Link name</td>
<td></td>
<td>WINNT</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>LAN destination address (hex)</td>
<td>9</td>
<td>08005AA5FAB9</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>Partner network ID</td>
<td></td>
<td>NETID</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>Partner node name</td>
<td></td>
<td>WINNTCP</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>LU name</td>
<td>5</td>
<td>WINNTLU</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>Alias (for remote LU name)</td>
<td></td>
<td>NTQMGR</td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>Mode</td>
<td>17</td>
<td>#INTER</td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>Remote transaction program name</td>
<td>7</td>
<td>MQSERIES</td>
<td></td>
</tr>
</tbody>
</table>

### Connection to an AIX system

The values in this section of the table must match those used in Table 21 on page 197, as indicated.

<table>
<thead>
<tr>
<th>ID</th>
<th>Parameter Name</th>
<th>Reference</th>
<th>Example Used</th>
<th>User Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>11</td>
<td>Link name</td>
<td></td>
<td>RS6000</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>LAN destination address (hex)</td>
<td>6</td>
<td>123456789012</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>Partner network ID</td>
<td>1</td>
<td>NETID</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>Partner node name</td>
<td>2</td>
<td>AIXPU</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>LU name</td>
<td>4</td>
<td>AIXLU</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>Alias (for remote LU name)</td>
<td>5</td>
<td>AIXQMGR</td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>Mode</td>
<td>17</td>
<td>#INTER</td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>Remote transaction program name</td>
<td>6</td>
<td>MQSERIES</td>
<td></td>
</tr>
</tbody>
</table>

### Connection to an HP-UX system

The values in this section of the table must match those used in Table 24 on page 219, as indicated.

<table>
<thead>
<tr>
<th>ID</th>
<th>Parameter Name</th>
<th>Reference</th>
<th>Example Used</th>
<th>User Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>11</td>
<td>Link name</td>
<td></td>
<td>HPUX</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>LAN destination address (hex)</td>
<td>8</td>
<td>100090DC2C7C</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>Partner network ID</td>
<td>4</td>
<td>NETID</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>Partner node name</td>
<td>2</td>
<td>HPUXPUP</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>LU name</td>
<td>5</td>
<td>HPUXLU</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>Alias (for remote LU name)</td>
<td>6</td>
<td>HPUXQMGR</td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>Mode</td>
<td>6</td>
<td>#INTER</td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>Remote transaction program name</td>
<td>7</td>
<td>MQSERIES</td>
<td></td>
</tr>
</tbody>
</table>
Table 15. Configuration worksheet for Communications Manager/2 (continued)

<table>
<thead>
<tr>
<th>ID</th>
<th>Parameter Name</th>
<th>Reference</th>
<th>Example Used</th>
<th>User Value</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><strong>Connection to an AT&amp;T GIS UNIX system</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>Link name</td>
<td>GIS</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>LAN destination address (hex)</td>
<td>8</td>
<td>10007038E6B</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>Partner network ID</td>
<td>2</td>
<td>NETID</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>Partner node name</td>
<td>3</td>
<td>GISPUI</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>LU name</td>
<td>4</td>
<td>GISLU</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>Alias (for remote LU name)</td>
<td></td>
<td>GISQMGR</td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>Mode</td>
<td>15</td>
<td>#INTER</td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>Remote transaction program name</td>
<td>5</td>
<td>MQSERIES</td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Connection to a Solaris system</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>Link name</td>
<td>SOLARIS</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>LAN destination address (hex)</td>
<td>5</td>
<td>08002071CC8A</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>Partner network ID</td>
<td>2</td>
<td>NETID</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>Partner node name</td>
<td>3</td>
<td>SOLARPU</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>LU name</td>
<td>7</td>
<td>SOLARLU</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>Alias (for remote LU name)</td>
<td></td>
<td>SOLQMGR</td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>Mode</td>
<td>17</td>
<td>#INTER</td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>Remote transaction program name</td>
<td>8</td>
<td>MQSERIES</td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Connection to an iSeries system</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>Link name</td>
<td>AS400</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>LAN destination address (hex)</td>
<td>4</td>
<td>10005A5962EF</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>Partner network ID</td>
<td>1</td>
<td>NETID</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>Partner node name</td>
<td>2</td>
<td>AS400PU</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>LU name</td>
<td>3</td>
<td>AS400LU</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>Alias (for remote LU name)</td>
<td></td>
<td>AS4QMGR</td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>Mode</td>
<td>17</td>
<td>#INTER</td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>Remote transaction program name</td>
<td>8</td>
<td>MQSERIES</td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Connection to a z/OS system without CICS</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>Link name</td>
<td>HOST0001</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>LAN destination address (hex)</td>
<td>8</td>
<td>400074511092</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>Partner network ID</td>
<td>2</td>
<td>NETID</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>Partner node name</td>
<td>3</td>
<td>MVSPU</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>LU name</td>
<td>4</td>
<td>MVSLU</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>Alias (for remote LU name)</td>
<td></td>
<td>MVSQMGR</td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>Mode</td>
<td>6</td>
<td>#INTER</td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>Remote transaction program name</td>
<td>7</td>
<td>MQSERIES</td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Connection to a z/OS system using a generic interface</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>Link name</td>
<td>HOST0001</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Table 15. Configuration worksheet for Communications Manager/2 (continued)  

<table>
<thead>
<tr>
<th>ID</th>
<th>Parameter Name</th>
<th>Reference</th>
<th>Example Used</th>
<th>User Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>12</td>
<td>LAN destination address (hex)</td>
<td>6</td>
<td>400074511092</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>Partner network ID</td>
<td>2</td>
<td>NETID</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>Partner node name</td>
<td>3</td>
<td>MVSPU</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>LU name</td>
<td>10</td>
<td>MVSQR</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>Alias (for remote LU name)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>Mode</td>
<td>6</td>
<td>#INTER</td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>Remote transaction program name</td>
<td>7</td>
<td>MQSERIES</td>
<td></td>
</tr>
</tbody>
</table>

Connection to a VSE/ESA system

The values in this section of the table must match those used in Table 50 on page 549 as indicated.

<table>
<thead>
<tr>
<th>ID</th>
<th>Parameter Name</th>
<th>Reference</th>
<th>Example Used</th>
<th>User Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>11</td>
<td>Link name</td>
<td></td>
<td>HOST0001</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>LAN destination address (hex)</td>
<td>5</td>
<td>400074511092</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>Partner network ID</td>
<td>1</td>
<td>NETID</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>Partner node name</td>
<td>2</td>
<td>VSEPU</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>LU name</td>
<td>3</td>
<td>VSELU</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>Alias (for remote LU name)</td>
<td></td>
<td>VSEQMGR</td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>Mode</td>
<td>6</td>
<td>#INTER</td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>Remote transaction program name</td>
<td>4</td>
<td>MQ01 MQ01</td>
<td></td>
</tr>
</tbody>
</table>

Explanation of terms

1. **Configuration name**
   This is the name of the OS/2 file that will hold the configuration.
   If you are adding to or modifying an existing configuration it will be the name previously specified.
   If you are creating a new configuration then you can specify any 8-character name that obeys the normal rules for file naming.

2. **Network ID**
   This is the unique ID of the network to which you are connected. It is an alphanumeric value and can be 1-8 characters long. The network ID works with the local node name to uniquely identify a system. Your network administrator will tell you the value.

3. **Local node name**
   This is the unique Control Point name for this workstation. Your network administrator will assign this to you.

4. **Local node ID (hex)**
   This is a unique identifier for this workstation. On other platforms it is often referred to as the exchange ID (XID). Your network administrator will assign this to you.

5. **Local node alias name**
   This is the name by which your local node will be known within this workstation. This value is not used elsewhere, but it is recommended that it be the same as the local node name.

6. **LU name (local)**
   An LU manages the exchange of data between systems. The local LU name is the name of the LU on your system. Your network administrator will assign this to you.
7 **Alias (for local LU name)**

The name by which your local LU will be known to your applications. You choose this name yourself. It can be 1-8 characters long. This value is used during WebSphere MQ configuration, when entries are added to the qm.ini file.

8 **Local transaction program (TP) name**

WebSphere MQ applications trying to converse with this workstation will specify a symbolic name for the program to be run at the receiving end. This will have been defined on the channel definition at the sender. The TP name is also used during WebSphere MQ configuration, when entries are added to the qm.ini file. For simplicity, wherever possible use a transaction program name of MQSERIES, or in the case of a connection to VSE/ESA, where the length is limited to 4 bytes, use MQTP.

See Table 13 on page 132 for more information.

9 **OS/2 program path and file name**

This is the path and name of the actual program to be run when a conversation has been initiated with this workstation. The example shown on the worksheet assumes that WebSphere MQ is installed in the default directory, c:\mqm. The configuration pairs this name with the symbolic name 8.

10 **LAN adapter address**

This is the address of your token-ring card. When using the default address, the exact value can be found in the LANTRAN.LOG file found in the \IBMCOM directory.

For example:

Adapter 0 is using node address 10005AFC5D83

11 **Link name**

This is a meaningful symbolic name by which the connection to a partner node is known. It is used only inside Communications Manager/2 setup and is specified by you. It can be 1-8 characters in length.

16 **Alias (for remote LU name)**

This is a value known only on this workstation and is used to represent the fully qualified partner LU name. You supply the value.

17 **Mode**

This is the name given to the set of parameters that control the APPC conversation. This name must be defined at each point in the network between the local and partner LUs. Your network administrator will assign this to you.
Establishing an LU 6.2 connection

This section describes how to establish an LU 6.2 connection using Communications Manager/2 Version 1.11. You may use any of the supported LU 6.2 products for this platform. The panels would look different from those shown but most of their content would be similar.

Defining local node characteristics

To set up the local node you need to perform these tasks:
1. Configure a DLC.
2. Configure the local node.
3. Add a local LU.
4. Add a transaction program definition.
5. Configure a mode.

To define the local node characteristics:
1. Start the Communications Manager/2 Installation and Setup program by typing CMSETUP on an OS/2 command line, and pressing Enter.
2. Press OK to continue.
3. Press Setup to create or modify a configuration.
4. Specify a name (up to 8-characters) for a new configuration file or select the one that you wish to update. The following examples guide you through the creation of a new configuration file. Treat them as a guide if you are modifying an existing configuration.
5. Press Yes.
6. Press Yes.
   In this example we set up connections using APPC over Token-ring. The
   following screen appears in two stages. When you first see it, highlight the line:
   APPC APIs through Token-ring

   The complete screen appears as shown below.

7. Press Configure....

Configuring a DLC

![Configuration screen]

1. Complete the values for Network ID (2) and Local node name (3).
   a.
   b. Select End node - no network node server.
   c. Click on Advanced.
   d. Select DLC - Token-ring or other LAN types and press Configure....
   e. Enter the value for C&SM LAN ID. This should be the same value as the
      Network ID entered earlier (2).
   f. Leave the remaining default values and press OK.

Configuring the local node

![Configuration screen]

1. Select SNA local node characteristics and press Configure....
2. Complete the value for Local node ID (hex) (4) using the values in your
   configuration worksheet.
3. Press Options...
4. Complete the value for Local node alias name (5) and press OK.
5. Press OK.
Using Communications Manager/2

6. Select SNA features and press Configure....

Adding a local LU

1. Select Local LUs and press Create....
2. Complete the fields LU name (6) and Alias (7).
3. Press OK.

Adding a transaction program definition

1. Select Transaction program definitions and press Create....
2. Complete the values for Transaction program (TP) name (8) and OS/2 program path and file name (9). If you are going to use Attach Manager to start the listener program, specify the Program parameter string, for example -m OS2 -n MQSERIES.
3. Press Continue....
4. Specify that the program is to be run in the Background and that it is to be Non-queued, Attach Manager started.
5. Press OK.
Configuring a mode

1. Select Modes and #INTER and press Change....
2. Ensure that the default values match those shown above and press Cancel.
3. Press Close to close the SNA Features List window.

Local configuration is complete.

The following sections describe how to create connections to other nodes.

Connecting to a peer system

To set up a connection to a peer system the steps are:
1. Adding a peer connection
2. Defining a partner LU

Start from the Communications Manager Profile List panel.

Select SNA connections and press Configure....
Adding a peer connection

1. Select To peer node and press Create....
2. Select Token-ring or other LAN types and press Continue....
3. Specify a Link name (11) and check Activate at startup.
4. Complete the fields LAN destination address (hex) (12), Partner network ID (13), and Partner node name (14).
5. Press Define Partner LUs....

Defining a partner LU

1. Complete the fields Network ID (13), LU name (15), and Alias (16).
2. Press Add.
3. Press OK.
4. Press OK.
5. Press Close.

If you have connections to make to other platforms repeat this section as appropriate.

If you have made all the connections you require proceed to “Verifying the configuration” on page 154 to complete Communications Manager/2 configuration.
Connecting to a host system

To set up a connection to a host system, for example z/OS or VSE/ESA, the steps are:

1. Adding a host connection
2. Defining a partner LU

Start from the Communications Manager Profile List panel.

Select SNA connections and press Configure....

Adding a host connection

1. Select To host and press Create....
2. Select Token-ring or other LAN types and press Continue....
3. Specify a Link name (11) and check Activate at startup.
4. Complete the fields LAN destination address (hex) (12), Partner network ID (13), and Partner node name (14).
5. Press Define Partner LUs....
Defining a partner LU

1. Complete the fields Network ID (13), LU name (15), and Alias (16).
2. Press Add.
3. Press OK.
4. Press OK.
5. Press Close.

If you have connections to make to other platforms, proceed to the appropriate section.

If you have made all the connections you require proceed to “Verifying the configuration” to complete Communications Manager/2 configuration.

Verifying the configuration

1. Press Close to close the Communications Manager Profile List panel.
2. Press Close.
3. Press Yes.
4. Press OK.
5. Press Close.
What next?

The LU 6.2 connection is now established. You are ready to complete the configuration. Go to "MQSeries for OS/2 Warp configuration" on page 159.

Establishing a TCP connection

1. From your desktop, open the TCP Icon View.

   ![TCP Icon View](image)

   The icons you see may vary from those shown above, depending on how you have installed the product.

2. Start the TCP Configuration program.

3. On the Network page, ensure that the IP Address and Subnet Mask fields have been completed.

4. Select the Autostart tab.

   ![TCP/IP Configuration](image)

5. Ensure that inetd is selected.

6. Select the Hostnames tab.

7. Ensure that This machine’s hostname, Local domain name, and Nameserver address have been completed.

8. Close the configuration notebook.

   **Note:** You may see a panel warning that the inetd superserver has been selected without selecting servers. Press No to indicate that you do not wish to correct this.
9. Press **Save** to save the changes made.

10. Verify that the \MPTN\ETC\SERVICES file, which is located on the drive where you installed IBM Multi-Protocol Transport Services (MPTS), contains the following line:

    MQSeries 1414/tcp # MQSeries Chan’l Listener

    If this line is not present, add it.

11. Verify that the file \MPTN\ETC\INETD.LST, located on the same drive contains the following line:

    MQSeries tcp c:\mqm\bin\amqcrsta [-m QName]

    If this line is not present, add it. Note that this assumes you have installed WebSphere MQ on the default drive and in the default directories.

12. (Re)start the inetd superserver, either by rebooting OS/2 or by stopping any existing inetd superserver and then entering `start inetd` on the command line.

**What next?**

The TCP connection is now established. You are ready to complete the configuration. Go to “MQSeries for OS/2 Warp configuration” on page 159.
Establishing a NetBIOS connection

A NetBIOS connection is initiated from a queue manager that uses the ConnectionName parameter on its channel definition to connect to a target listener. To set up a NetBIOS connection, follow these steps:

1. At each end of the channel specify the local NetBIOS name to be used by the WebSphere MQ channel processes, in the queue manager configuration file qm.ini or in the registry for Windows systems. For example, the NETBIOS stanza in qm.ini at the sending end might look like this:

   ```
   NETBIOS:
   LocalName=O2NETB1
   ```

   and at the receiving end:

   ```
   NETBIOS:
   LocalName=O2NETB2
   ```

2. At each end of the channel, look at the LANTRAN.LOG file in the \IBMCOM directory to see what LAN adapter number is used by NetBIOS on your system. If it is not 0, which WebSphere MQ uses by default, specify the correct value in the NETBIOS stanza of the qm.ini file or of the registry for Windows systems. For example:

   ```
   NETBIOS:
   AdapterNum=1
   ```

3. At the sending end, define a channel specifying the NetBIOS name being used at the other end of the channel. For example:

   ```
   DEFINE CHANNEL (OS2.WINNT.NET) CHLTYPE(SDR) +
   TRPTYPE(NETBIOS) +
   CONNAME(O2NetB2) +
   XMITQ(WINNT) +
   REPLACE
   ```

4. At the receiving end, define the corresponding receiver channel. For example:

   ```
   DEFINE CHANNEL (OS2.WINNT.NET) CHLTYPE(RCVR) +
   TRPTYPE(NETBIOS) +
   REPLACE
   ```

5. At the receiving end, start the WebSphere MQ listener:

   ```
   runmqlsr -t netbios
   ```

   Optionally you may specify values for the queue manager name, NetBIOS local name, number of sessions, number of names, and number of commands. See “Defining a NetBIOS connection” on page 135 for more information about setting up NetBIOS connections.

Establishing an SPX connection

This section discusses the following topics:

- IPX/SPX parameters
- SPX addressing
- Using the SPX KEEPALIVE option
- Receiving on SPX

IPX/SPX parameters

In most cases the default settings for the IPX/SPX parameters will suit your needs. However, you may need to modify some of them in your environment to tune its use for WebSphere MQ. The actual parameters and the method of changing them varies according to the platform and provider of SPX communications support. The
following sections describe some of these parameters, particularly those that may influence the operation of WebSphere MQ channels and client connections.

Please refer to the Novell Client for OS/2 documentation for full details of the use and setting of NET.CFG parameters.

The following IPX/SPX parameters can be added to the Novell NET.CFG file, and can affect WebSphere MQ SPX channels and client connections.

**IPX**

*sockets (range = 9 - 128, default 64)*

This specifies the total number of IPX sockets available. WebSphere MQ channels use this resource, so depending on the number of channels and the requirements of other IPX/SPX applications, you may need to increase this value.

**SPX**

*sessions (default 16)*

This specifies the total number of simultaneous SPX connections. Each WebSphere MQ channel or client connection uses one session. You may need to increase this value depending on the number of WebSphere MQ channels or client connections you need to run.

*retry count (default = 12)*

This controls the number of times an SPX session will resend unacknowledged packets. WebSphere MQ does not override this value.

*verify timeout, listen timeout, and abort timeout (milliseconds)*

These timeouts adjust the ‘Keepalive’ behavior. If an SPX sending end does not receive anything within the ‘verify timeout’ period, it sends a packet to the receiving end. It then waits for the duration of the ‘listen timeout’ for a response. If it still does not receive a response, it sends another packet and expects a response within the ‘abort timeout’ period.

**SPX addressing**

WebSphere MQ uses the SPX address of each machine to establish connectivity. The SPX address is specified in the following form:

\[
\text{network.node(socket)}
\]

where

*network* Is the 4-byte network address of the network on which the remote machine resides,

*node* Is the 6-byte node address, which is the LAN address of the LAN adapter in the remote machine, and

*socket* Is the 2-byte socket number on which the remote machine will listen.

The default socket number used by WebSphere MQ is 5E86. You can change the default socket number by specifying it in the queue manager configuration file qm.ini or the Windows systems registry. If you have taken the default options for installation, the qm.ini file for queue manager OS2 is found in `c:\mqm\qmgs\os2`. The lines in qm.ini might read:

```
SPX:
  SOCKET=n
```
For more information about values you can set in qm.ini, see Appendix D, "Configuration file stanzas for distributed queuing" on page 715.

The SPX address is later specified in the CONNAME parameter of the sender channel definition. If the WebSphere MQ systems being connected reside on the same network, the network address need not be specified. Similarly, if the remote system is listening on the default socket number (5E86), it need not be specified. A fully qualified SPX address in the CONNAME parameter would be:

\[
\text{CONNAME('network.node(socket)')}\]

but if the systems reside on the same network and the default socket number is used, the parameter would be:

\[
\text{CONNAME(node)}\]

A detailed example of the channel configuration parameters is given in "MQSeries for OS/2 Warp configuration".

**Using the SPX KEEPALIVE option**

If you want to use the KEEPALIVE option you need to add the following entry to your queue manager configuration file (qm.ini) or the Windows systems registry:

\[
\text{SPX: KeepAlive=yes}\]

You can use the timeout parameters described above to adjust the behavior of KEEPALIVE.

**Receiving on SPX**

Receiving channel programs are started in response to a startup request from the sending channel. To do this, a listener program has to be started to detect incoming network requests and start the associated channel.

You should use the WebSphere MQ listener.

**Using the WebSphere MQ listener**

To run the Listener supplied with WebSphere MQ, that starts new channels as threads, use the RUNMQLSR command. For example:

\[
\text{RUNMQLSR -t spx}\]

Optionally you may specify the queue manager name or the socket number if you are not using the defaults.

**MQSeries for OS/2 Warp configuration**

**Notes:**

1. You can use the sample program AMQSBCG to display, to the stdout spool, the contents and headers of all the messages in a queue. For example:

\[
\text{AMQSBCG q_name qmgr_name}\]

displays the contents of the queue \textit{q_name} defined in queue manager \textit{qmgr_name}.

2. The WebSphere MQ command used to start the TCP listener is:

\[
\text{runmqlsr -t tcp}\]

The listener enables receiver channels to start automatically in response to a start request from an inbound sender channel.
OS/2 configuration

3. You can start any channel from the command prompt using the command
   `runmqchl -c channel.name`

4. Error logs can be found in the directories `\mqm\qmgrs\qmgrname\errors`, `\mqm\qmgrs\@system\errors`, and `\mqm\errors`. In all cases, the most recent messages are at the end of `amqerr01.log`.

5. When you are using the command interpreter `runmqsc` to enter administration commands, a + at the end of a line indicates that the next line is a continuation. Ensure that there is a space between the last parameter and the continuation character.

Basic configuration

1. Create the queue manager from the OS/2 command line using the command:
   `crtmqm -u dlqname -q os2`

   where:
   - `os2` is the name of the queue manager
   - `-q` Indicates that this is to become the default queue manager
   - `-u dlqname` Specifies the name of the undeliverable message queue

   This command creates a queue manager and a set of default objects, and sets the DEADQ attribute of the queue manager.

2. For SNA channels add an LU 6.2 stanza to the queue manager’s qm.ini file:
   ```ini
   LU62:
   TPName=MQSERIES
   LocalLU=OS2QMGR
   ```

   If you have taken the default options for installation, the qm.ini file for queue manager `os2` is found in `c:\mqm\qmgrs\os2`.

3. Start the queue manager from the OS/2 command line using the command:
   `strmqm os2`

   where `os2` is the name given to the queue manager when it was created.

Channel configuration

The following sections detail the configuration to be performed on the OS/2 queue manager to implement the channel described in Figure 32 on page 101. In each case the MQSC command is shown.

Examples are given for connecting MQSeries for OS/2 Warp and WebSphere MQ for Windows. If you wish connect to another WebSphere MQ product use the appropriate set of values from the table in place of those for WebSphere MQ for Windows.

**Note:** The words in **bold** are user-specified and reflect the names of WebSphere MQ objects used throughout these examples. If you change the names used here, ensure that you also change the other references made to these objects throughout this book. All others are keywords and should be entered as shown.
Table 16. Configuration worksheet for MQSeries for OS/2 Warp

<table>
<thead>
<tr>
<th>Parameter Name</th>
<th>Reference</th>
<th>Example Used</th>
<th>User Value</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Definition for local node</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A  Queue Manager Name</td>
<td>OS2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>B  Local queue name</td>
<td>OS2.LOCALQ</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Connection to WebSphere MQ for Windows</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C  Remote queue manager name</td>
<td>WINNT</td>
<td></td>
<td></td>
</tr>
<tr>
<td>D  Remote queue name</td>
<td>WINNT.REMOTEQ</td>
<td></td>
<td></td>
</tr>
<tr>
<td>E  Queue name at remote system</td>
<td>WINNT.LOCALQ</td>
<td></td>
<td></td>
</tr>
<tr>
<td>F  Transmission queue name</td>
<td>WINNT</td>
<td></td>
<td></td>
</tr>
<tr>
<td>G  Sender (SNA) channel name</td>
<td>OS2.WINNT.SNA</td>
<td></td>
<td></td>
</tr>
<tr>
<td>H  Sender (TCP/IP) channel name</td>
<td>OS2.WINNT.TCP</td>
<td></td>
<td></td>
</tr>
<tr>
<td>I  Receiver (SNA) channel name</td>
<td>WINNT.OS2.SNA</td>
<td></td>
<td></td>
</tr>
<tr>
<td>J  Receiver (TCP/IP) channel name</td>
<td>WINNT.OS2.TCP</td>
<td></td>
<td></td>
</tr>
<tr>
<td>K  Receiver (NetBIOS) channel name</td>
<td>WINNT.OS2.NET</td>
<td></td>
<td></td>
</tr>
<tr>
<td>L  Receiver (SPX) channel name</td>
<td>WINNT.OS2.SPX</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Connection to WebSphere MQ for AIX</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C  Remote queue manager name</td>
<td>AIX</td>
<td></td>
<td></td>
</tr>
<tr>
<td>D  Remote queue name</td>
<td>AIX.REMOTEQ</td>
<td></td>
<td></td>
</tr>
<tr>
<td>E  Queue name at remote system</td>
<td>AIX.LOCALQ</td>
<td></td>
<td></td>
</tr>
<tr>
<td>F  Transmission queue name</td>
<td>AIX</td>
<td></td>
<td></td>
</tr>
<tr>
<td>G  Sender (SNA) channel name</td>
<td>OS2.AIX.SNA</td>
<td></td>
<td></td>
</tr>
<tr>
<td>H  Sender (TCP/IP) channel name</td>
<td>OS2.AIX.TCP</td>
<td></td>
<td></td>
</tr>
<tr>
<td>I  Receiver (SNA) channel name</td>
<td>AIX.OS2.SNA</td>
<td></td>
<td></td>
</tr>
<tr>
<td>J  Receiver (TCP/IP) channel name</td>
<td>AIX.OS2.TCP</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Connection to Compaq Tru64 UNIX</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C  Remote queue manager name</td>
<td>DECUX</td>
<td></td>
<td></td>
</tr>
<tr>
<td>D  Remote queue name</td>
<td>DECUX.REMOTEQ</td>
<td></td>
<td></td>
</tr>
<tr>
<td>E  Queue name at remote system</td>
<td>DECUX.LOCALQ</td>
<td></td>
<td></td>
</tr>
<tr>
<td>F  Transmission queue name</td>
<td>DECUX</td>
<td></td>
<td></td>
</tr>
<tr>
<td>H  Receiver (TCP) channel name</td>
<td>DECUX.OS2.TCP</td>
<td></td>
<td></td>
</tr>
<tr>
<td>J  Receiver (TCP) channel name</td>
<td>OS2.DECUX.TCP</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Connection to WebSphere MQ for HP-UX</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C  Remote queue manager name</td>
<td>HPUX</td>
<td></td>
<td></td>
</tr>
<tr>
<td>D  Remote queue name</td>
<td>HPUX.REMOTEQ</td>
<td></td>
<td></td>
</tr>
<tr>
<td>E  Queue name at remote system</td>
<td>HPUX.LOCALQ</td>
<td></td>
<td></td>
</tr>
<tr>
<td>F  Transmission queue name</td>
<td>HPUX</td>
<td></td>
<td></td>
</tr>
<tr>
<td>G  Sender (SNA) channel name</td>
<td>OS2.HPUX.SNA</td>
<td></td>
<td></td>
</tr>
<tr>
<td>H  Sender (TCP) channel name</td>
<td>OS2.HPUX.TCP</td>
<td></td>
<td></td>
</tr>
<tr>
<td>I  Receiver (SNA) channel name</td>
<td>HPUX.OS2.SNA</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### OS/2 configuration

#### Table 16. Configuration worksheet for MQSeries for OS/2 Warp (continued)

<table>
<thead>
<tr>
<th>Parameter Name</th>
<th>Reference</th>
<th>Example Used</th>
<th>User Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>J Receiver (TCP) channel name</td>
<td>H</td>
<td>HPUX.OS2.TCP</td>
<td></td>
</tr>
</tbody>
</table>

**Connection to MQSeries for AT&T GIS UNIX**

The values in this section of the table must match those used in Table 27 on page 253 as indicated.

<table>
<thead>
<tr>
<th>C Remote queue manager name</th>
<th>A</th>
<th>GIS</th>
</tr>
</thead>
<tbody>
<tr>
<td>D Remote queue name</td>
<td></td>
<td>GIS.REMOTEQ</td>
</tr>
<tr>
<td>E Queue name at remote system</td>
<td>B</td>
<td>GIS.LOCALQ</td>
</tr>
<tr>
<td>F Transmission queue name</td>
<td></td>
<td>GIS</td>
</tr>
<tr>
<td>G Sender (SNA) channel name</td>
<td></td>
<td>OS2.GIS.SNA</td>
</tr>
<tr>
<td>H Sender (TCP) channel name</td>
<td></td>
<td>OS2.GIS.TCP</td>
</tr>
<tr>
<td>I Receiver (SNA) channel name</td>
<td>C</td>
<td>GIS.OS2.SNA</td>
</tr>
<tr>
<td>J Receiver (TCP) channel name</td>
<td>H</td>
<td>GIS.OS2.TCP</td>
</tr>
</tbody>
</table>

**Connection to WebSphere MQ for Solaris**

The values in this section of the table must match those used in Table 30 on page 290 as indicated.

<table>
<thead>
<tr>
<th>C Remote queue manager name</th>
<th>D</th>
<th>SOLARIS</th>
</tr>
</thead>
<tbody>
<tr>
<td>D Remote queue name</td>
<td></td>
<td>SOLARIS.REMOTEQ</td>
</tr>
<tr>
<td>E Queue name at remote system</td>
<td>B</td>
<td>SOLARIS.LOCALQ</td>
</tr>
<tr>
<td>F Transmission queue name</td>
<td></td>
<td>SOLARIS</td>
</tr>
<tr>
<td>G Sender (SNA) channel name</td>
<td></td>
<td>OS2.SOLARIS.SNA</td>
</tr>
<tr>
<td>H Sender (TCP/IP) channel name</td>
<td></td>
<td>OS2.SOLARIS.TCP</td>
</tr>
<tr>
<td>I Receiver (SNA) channel name</td>
<td>C</td>
<td>SOLARIS.OS2.SNA</td>
</tr>
<tr>
<td>J Receiver (TCP/IP) channel name</td>
<td>H</td>
<td>SOLARIS.OS2.TCP</td>
</tr>
</tbody>
</table>

**Connection to WebSphere MQ for iSeries**

The values in this section of the table must match those used in Table 49 on page 535 as indicated.

<table>
<thead>
<tr>
<th>C Remote queue manager name</th>
<th>D</th>
<th>AS400</th>
</tr>
</thead>
<tbody>
<tr>
<td>D Remote queue name</td>
<td></td>
<td>AS400.REMOTEQ</td>
</tr>
<tr>
<td>E Queue name at remote system</td>
<td>B</td>
<td>AS400.LOCALQ</td>
</tr>
<tr>
<td>F Transmission queue name</td>
<td></td>
<td>AS400</td>
</tr>
<tr>
<td>G Sender (SNA) channel name</td>
<td></td>
<td>OS2.AS400.SNA</td>
</tr>
<tr>
<td>H Sender (TCP/IP) channel name</td>
<td></td>
<td>OS2.AS400.TCP</td>
</tr>
<tr>
<td>I Receiver (SNA) channel name</td>
<td>C</td>
<td>AS400.OS2.SNA</td>
</tr>
<tr>
<td>J Receiver (TCP) channel name</td>
<td>H</td>
<td>AS400.OS2.TCP</td>
</tr>
</tbody>
</table>

**Connection to WebSphere MQ for z/OS without CICS**

The values in this section of the table must match those used in Table 34 on page 372 as indicated.

<table>
<thead>
<tr>
<th>C Remote queue manager name</th>
<th>D</th>
<th>MVS</th>
</tr>
</thead>
<tbody>
<tr>
<td>D Remote queue name</td>
<td></td>
<td>MVS.REMOTEQ</td>
</tr>
<tr>
<td>E Queue name at remote system</td>
<td>B</td>
<td>MVS.LOCALQ</td>
</tr>
<tr>
<td>F Transmission queue name</td>
<td></td>
<td>MVS</td>
</tr>
<tr>
<td>G Sender (SNA) channel name</td>
<td></td>
<td>OS2.MVS.SNA</td>
</tr>
<tr>
<td>H Sender (TCP) channel name</td>
<td></td>
<td>OS2.MVS.TCP</td>
</tr>
<tr>
<td>I Receiver (SNA) channel name</td>
<td>C</td>
<td>MVS.OS2.SNA</td>
</tr>
<tr>
<td>J Receiver (TCP) channel name</td>
<td>H</td>
<td>MVS.OS2.TCP</td>
</tr>
</tbody>
</table>

**Connection to WebSphere MQ for z/OS using queue sharing groups**

The values in this section of the table must match those used in Table 43 on page 454 as indicated.
Table 16. Configuration worksheet for MQSeries for OS/2 Warp (continued)

<table>
<thead>
<tr>
<th>Parameter Name</th>
<th>Reference</th>
<th>Example Used</th>
<th>User Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>C</td>
<td></td>
<td>QSG</td>
<td></td>
</tr>
<tr>
<td>D</td>
<td></td>
<td>QSG.REMOTEQ</td>
<td></td>
</tr>
<tr>
<td>E</td>
<td></td>
<td>QSG.SHAREDQ</td>
<td></td>
</tr>
<tr>
<td>F</td>
<td></td>
<td>QSG</td>
<td></td>
</tr>
<tr>
<td>G</td>
<td></td>
<td>OS2.QSG.SNA</td>
<td></td>
</tr>
<tr>
<td>H</td>
<td></td>
<td>OS2.QSG.TCP</td>
<td></td>
</tr>
<tr>
<td>I</td>
<td></td>
<td>QSG.OS2.SNA</td>
<td></td>
</tr>
<tr>
<td>J</td>
<td></td>
<td>QSG.OS2.TCP</td>
<td></td>
</tr>
</tbody>
</table>

Connection to MQSeries for VSE/ESA

The values in this section of the table must match those used in Table 51 on page 554 as indicated.

<table>
<thead>
<tr>
<th>Parameter Name</th>
<th>Reference</th>
<th>Example Used</th>
<th>User Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>C</td>
<td></td>
<td>VSE</td>
<td></td>
</tr>
<tr>
<td>D</td>
<td></td>
<td>VSE.REMOTEQ</td>
<td></td>
</tr>
<tr>
<td>E</td>
<td></td>
<td>VSE.LOCALQ</td>
<td></td>
</tr>
<tr>
<td>F</td>
<td></td>
<td>VSE</td>
<td></td>
</tr>
<tr>
<td>G</td>
<td></td>
<td>OS2.VSE.SNA</td>
<td></td>
</tr>
<tr>
<td>I</td>
<td></td>
<td>VSE.OS2.SNA</td>
<td></td>
</tr>
</tbody>
</table>

MQSeries for OS/2 Warp sender-channel definitions using SNA

```text
def ql (WINNT) usage(xmitq) + replace
def qr (WINNT.REMOTEQ) + rname(WINNT.LOCALQ) + rgmname(WINNT) + xmitq(WINNT) + replace
def chl (OS2.WINNT.SNA) chltype(sdr) + trptype(lu62) + connname(’NETID.WINNTLU’) + xmitq(WINNT) + modename(‘#INTER’) + tpname(’MQSERIES’) + replace
```

MQSeries for OS/2 Warp receiver-channel definitions using SNA

```text
def ql (OS2.LOCALQ) replace
def chl (WINNT.OS2.SNA) chltype(rcvr) + trptype(lu62) + replace
```

MQSeries for OS/2 Warp sender-channel definitions using TCP

```text
def ql (WINNT) usage(xmitq) + replace
def qr (WINNT.REMOTEQ) + rname(WINNT.LOCALQ) + rgmname(WINNT) + xmitq(WINNT) + replace
def chl (OS2.WINNT.TCP) chltype(sdr) +
```
OS/2 configuration

trptype(tcp) +
conname(remote_tcpip_hostname) +
xmitq(WINNT) +
replace

MQSeries for OS/2 Warp receiver-channel definitions using TCP/IP

def q1 (OS2.LOCALQ) replace

def chl (WINNT.OS2.TCP) chltype(rcvr) +
trptype(tcp) +
replace

MQSeries for OS/2 Warp sender-channel definitions using NetBIOS

def q1 (WINNT) +
usage(xmitq) +
replace

def qr (WINNT.REMOTEQ) +
  rname(WINNT.LOCALQ) +
  rqmname(WINNT) +
  xmitq(WINNT) +
  replace

def chl (OS2.WINNT.NET) chltype(sdr) +
trptype(netbios) +
conname(remote NetBIOS name) +
xmitq(WINNT) +
replace

MQSeries for OS/2 Warp receiver-channel definitions using NetBIOS

def q1 (OS2.LOCALQ) replace

def chl (WINNT.OS2.NET) chltype(rcvr) +
trptype(netbios) +
replace

MQSeries for OS/2 Warp sender-channel definitions using IPX/SPX

def q1 (WINNT) +
usage(xmitq) +
replace

def qr (WINNT.REMOTEQ) +
  rname(WINNT.LOCALQ) +
  rqmname(WINNT) +
  xmitq(WINNT) +
  replace

def chl (OS2.WINNT.SPX) chltype(sdr) +
trptype(spx) +
conname('network.node(socket)') +
xmitq(WINNT) +
replace

MQSeries for OS/2 Warp receiver-channel definitions using IPX/SPX

def q1 (OS2.LOCALQ) replace

def chl (WINNT.OS2.SPX) chltype(rcvr) +
trptype(spx) +
replace
Running channels as processes or threads

MQSeries for OS/2 Warp provides the flexibility to run sender channels as OS/2 processes or OS/2 threads. This is specified in the MCATYPE parameter on the sender channel definition. Each installation should select the type appropriate for their application and configuration. Factors affecting this choice are discussed below.

Most installations will select to run their sender channels as threads, because the virtual and real memory required to support a large number of concurrent channel connections will be reduced. When the WebSphere MQ listener process (started via the RUNMQLSR command) exhausts the available private memory needed, an additional listener process will need to be started to support more channel connections. When each channel runs as a process, additional processes are automatically started, avoiding the out-of-memory condition.

If all channels are run as threads under one WebSphere MQ listener, a failure of the listener for any reason will cause all channel connections to be temporarily lost. This can be prevented by balancing the threaded channel connections across two or more listener processes, thus enabling other connections to keep running. If each sender channel is run as a separate process, the failure of the listener for that process will affect only that specific channel connection.

A NetBIOS connection needs a separate process for the Message Channel Agent. Therefore, before you can issue a START CHANNEL command, you must start the channel initiator, or you may start a channel using the RUNMQCHL command.
DQM in distributed platforms
Chapter 12. Example configuration - IBM WebSphere MQ for Windows

This chapter gives an example of how to set up communication links from WebSphere MQ for Windows to WebSphere MQ products on the following platforms:
- OS/2
- AIX
- Compaq Tru64 UNIX
- HP-UX
- AT&T GIS UNIX²
- Solaris
- OS/400
- z/OS without CICS
- VSE/ESA

This chapter first describes the parameters needed for an LU 6.2 connection, then it guides you through the following tasks:
- “Establishing an LU 6.2 connection” on page 172
- “Establishing a TCP connection” on page 180
- “Establishing a NetBIOS connection” on page 180
- “Establishing an SPX connection” on page 181

Once the connection is established, you need to define some channels to complete the configuration. This is described in “WebSphere MQ for Windows configuration” on page 183.

See Chapter 7, “Example configuration chapters in this book” on page 101 for background information about this chapter and how to use it.

Configuration parameters for an LU 6.2 connection

Table 17 on page 168 presents a worksheet listing all the parameters needed to set up communication from Windows to one of the other WebSphere MQ platforms. The worksheet shows examples of the parameters, which have been tested in a working environment, and leaves space for you to fill in your own values. An explanation of the parameter names follows the worksheet. Use the worksheet in this chapter in conjunction with the worksheet in the chapter for the platform to which you are connecting.

The steps required to set up an LU 6.2 connection are described, with numbered cross references to the parameters on the worksheet. These steps are:
- “Configuring the local node” on page 172
- “Adding a connection” on page 174
- “Adding a partner” on page 176
- “Adding a CPI-C entry” on page 177
- “Configuring an invokable TP” on page 177

2. This platform has become NCR UNIX SVR4 MP-RAS, R3.0
Windows systems and LU 6.2

Configuration worksheet

Use this worksheet to record the values you use for your configuration. Where numbers appear in the Reference column they indicate that the value must match that in the appropriate worksheet elsewhere in this book. The examples that follow in this chapter refer back to the values in the ID column. The entries in the Parameter Name column are explained in “Explanation of terms” on page 170.

Table 17. Configuration worksheet for IBM Communications Server for Windows systems

<table>
<thead>
<tr>
<th>ID</th>
<th>Parameter Name</th>
<th>Reference</th>
<th>Example Used</th>
<th>User Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Configuration name</td>
<td>NTCONFIG</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Network Name</td>
<td>NETID</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Control Point Name</td>
<td>WINNTCP</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Local Node ID (hex)</td>
<td>05D 30F65</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>LU Name (local)</td>
<td>WINNTLU</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>LU Alias (local)</td>
<td>NTQMGR</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>TP Name</td>
<td>MQSERIES</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>Command line</td>
<td><code>c:\mqm\bin\amqcrs6a.exe</code></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>LAN adapter address</td>
<td>08005AA5FAB9</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Definition for local node

1. Configuration name  NTCONFIG
2. Network Name  NETID
3. Control Point Name  WINNTCP
4. Local Node ID (hex)  05D 30F65
5. LU Name (local)  WINNTLU
6. LU Alias (local)  NTQMGR
7. TP Name  MQSERIES
8. Command line  `c:\mqm\bin\amqcrs6a.exe`
9. LAN adapter address  08005AA5FAB9

Connection to an OS/2 system

The values in this section of the table must match those used in Table 15 on page 144, as indicated.

<table>
<thead>
<tr>
<th>ID</th>
<th>Parameter Name</th>
<th>Reference</th>
<th>Example Used</th>
<th>User Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>Connection</td>
<td>OS2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>Remote Network Address</td>
<td>10  10005AFC3D83</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>Network Name</td>
<td>2</td>
<td>NETID</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>Control Point Name</td>
<td>3</td>
<td>OS2PU</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>Remote Node ID</td>
<td>4</td>
<td>05D 12345</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>LU Alias (remote)</td>
<td>OS2QMGR</td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>LU Name</td>
<td>6</td>
<td>OS2LU</td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>Mode</td>
<td>17</td>
<td>#INTER</td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>CPI-C Name</td>
<td>18</td>
<td>OS2CPIC</td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>Partner TP Name</td>
<td>19</td>
<td>MQSERIES</td>
<td></td>
</tr>
</tbody>
</table>

Connection to an AIX system

The values in this section of the table must match those used in Table 21 on page 197, as indicated.

<table>
<thead>
<tr>
<th>ID</th>
<th>Parameter Name</th>
<th>Reference</th>
<th>Example Used</th>
<th>User Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>Connection</td>
<td>AIX</td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>Remote Network Address</td>
<td>8</td>
<td>123456789012</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>Network Name</td>
<td>1</td>
<td>NETID</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>Control Point Name</td>
<td>2</td>
<td>AIXPU</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>Remote Node ID</td>
<td>3</td>
<td>071 23456</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>LU Alias (remote)</td>
<td>AIXQMGR</td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>LU Name</td>
<td>4</td>
<td>AIXLU</td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>Mode</td>
<td>14</td>
<td>#INTER</td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>CPI-C Name</td>
<td>18</td>
<td>AIXCPIC</td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>Partner TP Name</td>
<td>19</td>
<td>MQSERIES</td>
<td></td>
</tr>
</tbody>
</table>

Connection to an HP-UX system

The values in this section of the table must match those used in Table 24 on page 219, as indicated.

<table>
<thead>
<tr>
<th>ID</th>
<th>Parameter Name</th>
<th>Reference</th>
<th>Example Used</th>
<th>User Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>Connection</td>
<td>HPUX</td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>Remote Network Address</td>
<td>8</td>
<td>100090DC2C7C</td>
<td></td>
</tr>
</tbody>
</table>
Table 17. Configuration worksheet for IBM Communications Server for Windows systems (continued)

<table>
<thead>
<tr>
<th>ID</th>
<th>Parameter Name</th>
<th>Reference</th>
<th>Example Used</th>
<th>User Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>12</td>
<td>Network Name</td>
<td>4</td>
<td>NETID</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>Control Point Name</td>
<td>2</td>
<td>HPUXPU</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>Remote Node ID</td>
<td>3</td>
<td>05D 54321</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>LU Alias (remote)</td>
<td></td>
<td>HPUXQMGR</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>LU Name</td>
<td>5</td>
<td>HPUXLU</td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>Mode</td>
<td>17</td>
<td>#INTER</td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>CPI-C Name</td>
<td>7</td>
<td>HPUXCPIC</td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>Partner TP Name</td>
<td></td>
<td>MQSERIES</td>
<td></td>
</tr>
</tbody>
</table>

**Connection to an AT&T GIS UNIX system**

The values in this section of the table must match those used in Table 26 on page 243, as indicated.

| 10 | Connection                         | GIS       |             |            |
| 11 | Remote Network Address             | 8         | 10007038E86B|            |
| 12 | Network Name                       | 2         | NETID       |            |
| 13 | Control Point Name                 | 3         | GISPU       |            |
| 14 | Remote Node ID                     | 9         | 03E 00018   |            |
| 15 | LU Alias (remote)                  |           | GISQMGR     |            |
| 16 | LU Name                            | 4         | GISLU       |            |
| 17 | Mode                               | 15        | #INTER      |            |
| 18 | CPI-C Name                         | 7         | GISCPLIC    |            |
| 19 | Partner TP Name                    | 5         | MQSERIES    |            |

**Connection to a Solaris system**

The values in this section of the table must match those used in Table 28 on page 258, as indicated.

| 10 | Connection                         | SOLARIS   |             |            |
| 11 | Remote Network Address             | 5         | 08002071CC8A|            |
| 12 | Network Name                       | 2         | NETID       |            |
| 13 | Control Point Name                 | 3         | SOLARPU     |            |
| 14 | Remote Node ID                     | 6         | 05D 31OD6   |            |
| 15 | LU Alias (remote)                  |           | SOLARQMGR   |            |
| 16 | LU Name                            | 7         | SOLARLU     |            |
| 17 | Mode                               | 17        | #INTER      |            |
| 18 | CPI-C Name                         | 8         | SOLCPIC     |            |
| 19 | Partner TP Name                    | 8         | MQSERIES    |            |

**Connection to an iSeries system**

The values in this section of the table must match those used in Table 48 on page 521, as indicated.

| 10 | Connection                         | AS400     |             |            |
| 11 | Remote Network Address             | 4         | 10005A5962EF|            |
| 12 | Network Name                       | 1         | NETID       |            |
| 13 | Control Point Name                 | 2         | AS400PU     |            |
| 14 | Remote Node ID                     |           | AS400QMGR   |            |
| 15 | LU Alias (remote)                  |           | AS400LU     |            |
| 16 | LU Name                            | 3         | AS4CPIC     |            |
| 17 | Mode                               | 17        | #INTER      |            |
| 18 | CPI-C Name                         | 8         | MQSERIES    |            |
| 19 | Partner TP Name                    |           |             |            |
### Windows systems and LU 6.2

**Table 17. Configuration worksheet for IBM Communications Server for Windows systems (continued)**

<table>
<thead>
<tr>
<th>ID</th>
<th>Parameter Name</th>
<th>Reference</th>
<th>Example Used</th>
<th>User Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>Connection</td>
<td></td>
<td>MVS</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>Remote Network Address</td>
<td>3</td>
<td>40074511092</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>Network Name</td>
<td>2</td>
<td>NETID</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>Control Point Name</td>
<td>3</td>
<td>MVSPU</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>Remote Node ID</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>LU Alias (remote)</td>
<td>MVSQMGR</td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>LU Name</td>
<td>4</td>
<td>MVSLU</td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>Mode</td>
<td>10</td>
<td>#INTER</td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>CPI-C Name</td>
<td></td>
<td>MVSCPIC</td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>Partner TP Name</td>
<td>7</td>
<td>MQSERIES</td>
<td></td>
</tr>
</tbody>
</table>

### Connection to a z/OS system without CICS

The values in this section of the table must match those used in Table 33 on page 366, as indicated.

<table>
<thead>
<tr>
<th>ID</th>
<th>Parameter Name</th>
<th>Reference</th>
<th>Example Used</th>
<th>User Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>Connection</td>
<td></td>
<td>MVS</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>Remote Network Address</td>
<td>3</td>
<td>40074511092</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>Network Name</td>
<td>2</td>
<td>NETID</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>Control Point Name</td>
<td>3</td>
<td>MVSPU</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>Remote Node ID</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>LU Alias (remote)</td>
<td>MVSQMGR</td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>LU Name</td>
<td>4</td>
<td>MVSLU</td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>Mode</td>
<td>10</td>
<td>#INTER</td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>CPI-C Name</td>
<td></td>
<td>MVSCPIC</td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>Partner TP Name</td>
<td>7</td>
<td>MQSERIES</td>
<td></td>
</tr>
</tbody>
</table>

### Connection to a z/OS system using a generic interface

The values in this section of the table must match those used in Table 33 on page 366, as indicated.

<table>
<thead>
<tr>
<th>ID</th>
<th>Parameter Name</th>
<th>Reference</th>
<th>Example Used</th>
<th>User Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>Connection</td>
<td></td>
<td>MVS</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>Remote Network Address</td>
<td>3</td>
<td>40074511092</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>Network Name</td>
<td>2</td>
<td>NETID</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>Control Point Name</td>
<td>3</td>
<td>MVSPU</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>Remote Node ID</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>LU Alias (remote)</td>
<td>MVSQMGR</td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>LU Name</td>
<td>18</td>
<td>MVSLU</td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>Mode</td>
<td>6</td>
<td>#INTER</td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>CPI-C Name</td>
<td></td>
<td>MVSCPIC</td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>Partner TP Name</td>
<td>7</td>
<td>MQSERIES</td>
<td></td>
</tr>
</tbody>
</table>

### Connection to a VSE/ESA system

The values in this section of the table must match those used in Table 50 on page 549, as indicated.

<table>
<thead>
<tr>
<th>ID</th>
<th>Parameter Name</th>
<th>Reference</th>
<th>Example Used</th>
<th>User Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>Connection</td>
<td></td>
<td>MVS</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>Remote Network Address</td>
<td>3</td>
<td>40074511092</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>Network Name</td>
<td>1</td>
<td>NETID</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>Control Point Name</td>
<td>2</td>
<td>VSEPU</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>Remote Node ID</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>LU Alias (remote)</td>
<td>VSEQMGR</td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>LU Name</td>
<td>3</td>
<td>VSELU</td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>Mode</td>
<td></td>
<td>#INTER</td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>CPI-C Name</td>
<td></td>
<td>VSECXPIC</td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>Partner TP Name</td>
<td>4</td>
<td>MQ01</td>
<td>MQ01</td>
</tr>
</tbody>
</table>

### Explanation of terms

1. **Configuration Name**
   
   This is the name of the file in which the Communications Server configuration is saved.

2. **Network Name**
   
   This is the unique ID of the network to which you are connected. It is an alphanumeric value and can be 1-8 characters long. The network name
works with the Control Point Name to uniquely identify a system. Your network administrator will tell you the value.

3 Control Point Name
In Advanced Peer-to-Peer Networking® (APPN®), a control point is responsible for managing a node and its resources. A control point is also a logical unit (LU). The Control Point Name is the name of the LU and is assigned to your system by the network administrator.

4 Local Node ID (hex)
Some SNA products require partner systems to specify a node identifier that uniquely identifies their workstation. The two systems exchange this node identifier in a message unit called the exchange identifier (XID). Your network administrator will assign this ID for you.

5 LU Name (local)
A logical unit (LU) is software that serves as an interface or translator between a transaction program and the network. An LU manages the exchange of data between transaction programs. The local LU Name is the name of the LU on your workstation. Your network administrator will assign this to you.

6 LU Alias (local)
The name by which your local LU will be known to your applications. You choose this name yourself. It can be 1-8 characters long.

7 TP Name
WebSphere MQ applications trying to converse with your workstation specify a symbolic name for the program that is to start running. This will have been defined on the channel definition at the sender. For simplicity, wherever possible use a transaction program name of MQSERIES, or in the case of a connection to VSE/ESA, where the length is limited to 4 bytes, use MQTP.
See Table 13 on page 132 for more information.

8 Command line
This is the path and name of the actual program to be run when a conversation has been initiated with your workstation. The example shown on the worksheet assumes that WebSphere MQ is installed in the default directory, c:\mqm. The configuration pairs this name with the symbolic name when you use TPSETUP (which is part of the SNA Server software developers kit).

9 LAN adapter address
This is the address of your token-ring card. To discover this type net config server at a command prompt. The address appears in the output. For example:
Server is active on 08005AA5FA89

10 Connection
This is a meaningful symbolic name by which the connection to a partner node is known. It is used only within SNA Server administration and is specified by you.

15 LU Alias (remote)
This is a value known only in this server and is used to represent the fully qualified partner LU name. You supply the value.

17 Mode
This is the name given to the set of parameters that control the APPC
Windows systems and LU 6.2

An entry with this name and a similar set of parameters must be defined at each partner system. Your network administrator will tell you this name.

**CPI-C Name**

This is the name given to a locally held definition of a partner application. You supply the name and it must be unique within this server. The name is specified in the CONNAME attribute of the WebSphere MQ sender channel definition.

---

**Establishing an LU 6.2 connection**

This section describes how to establish an LU 6.2 connection using IBM Communications Server for Windows NT, Version 5.0. You may use any of the supported LU 6.2 products for this platform. The panels of other products will not be identical to those shown here, but most of their content will be similar.

**Configuring the local node**

To configure the local node, follow these steps:

1. From the Scenarios pull-down of the Communications Server SNA Node Configuration window, select the CPI-C, APPC or 5250 Emulation scenario.

   ![WebSphere MQ Intercommunication](image)

   The CPI-C, APPC or 5250 Emulation scenario window is displayed.

2. Click on Configure Node, then click on New. The Define the Node property sheet is displayed.
3. In the **Fully qualified CP name** field on the Basic page, enter the unique ID of the network to which you are connected (2) and the control point name (3). Click on **OK** to continue.

4. From the SNA Node Configuration window, click on **Configure Local LU 6.2**, then click on **New**. The Define a Local LU 6.2 window is displayed.
5. In the Local LU name field on the Basic page, enter the name of the LU on your workstation (\SF5800005\SF590000). In the Local LU alias field, enter the name by which your local LU will be known to your applications (\SF5800006\SF590000). Click on OK to continue.

Adding a connection

To add a connection, follow these steps:

1. From the SNA Node Configuration window, select Configure Devices, select LAN as the DLC type, then click on New. The Define a LAN Device property sheet is displayed.

2. If you have the LLC2 protocol installed with Communications Server for Windows NT, the Adapter number list box lists the available LAN adapters. See the help file INLLC40.HLP (Windows NT 4.0) or INLLC35.HLP (Windows NT 3.51) in the Communications Server installation directory for LLC2 installation instructions.

3. The default values displayed on the Define a LAN Device Basic page may be accepted. Click on OK to continue.

4. From the SNA Node Configuration window, select Configure Connections, select LAN as the DLC type, then click on New. The Define a LAN Connection property sheet is displayed.
5. In the **Destination address** field on the Basic page, enter the LAN address of the system to which you are connecting (1005AFC3D8F4). Select the Advanced page.

6. In the **Block ID** field on the Advanced page, enter the local node ID (hex) (04). Select the Security page.
7. In the **Adjacent CP name** field on the Security page, enter the network name and control point name of the remote node (SF58000012/SF59000013). In the **Adjacent CP type** field, enter APPN Node. You do not need to complete the **Adjacent node ID** field for a peer-to-peer connection. Click on **OK** to continue. Take note of the default link name used to identify this new definition (for example, LINK0000).

### Adding a partner

To add a partner LU definition, follow these steps:

1. From the SNA Node Configuration window, select **Configure Partner LU 6.2**, then click on **New**. The Define a Partner LU 6.2 property sheet is displayed.

2. In the **Partner LU name** field on the Basic page, enter the network name (SF58000012) and LU name of the remote system (SF58000016). In the **Partner LU alias** field, enter the remote LU alias (SF58000015). In the **Fully qualified CP name** fields, enter the network name and control point name of the remote system (SF58000012 and SF58000013). Click on **OK** to continue.
Adding a CPI-C entry

To add a CPI-C Side information entry, follow these steps:

1. From the SNA Node Configuration window, select Configure CPI-C Side Information, then click on New. The Define a CPI-C Side Information property sheet is displayed.

![Define CPI-C Side Information](image)

2. In the Symbolic destination name field of the Basic page, enter the CPI-C name (18). In the Mode name field, enter the mode value (17). Enter either a fully qualified partner LU name (12, 16) or a partner LU alias (15) depending on what you choose in the CPI-C Side Information property sheet. In the TP name field, enter the partner TP name (19). Click on OK to continue.

Configuring an invokable TP

To add a Transaction Program (TP) definition, follow these steps:

1. From the SNA Node Configuration window, select Configure Transaction Programs, then click on New. The Define a Transaction Program property sheet is displayed.
Using IBM Communications Server

2. In the **TP name** field on the Basic page, enter the transaction program name (**7**). In the **Complete pathname** field, enter the actual path and name of the program that will be run when a conversation is initiated with your workstation (**8**). When you are happy with the settings, click on **OK** to continue.

3. In order to be able to stop the WebSphere MQ Transaction Program, you need to start it in one of the following ways:
   
a. Check **Service TP** on the Basic page. This starts the TP programs at Windows NT startup and will run the programs under the system user ID.

   b. Check **Dynamically loaded** on the Advanced page. This dynamically loads and starts the programs as and when incoming SNA conversation requests arrive. It will run the programs under the same user ID as the rest of WebSphere MQ.

   **Note:** To use dynamic loading, it is necessary to vary the user ID under which the WebSphere MQ SNA Transaction program runs. To do this, set the Attach Manager to run under the desired user context by modifying the startup parameters within the Control Panel in the Services applet for the AppnNode service.

   c. Issue the WebSphere MQ command, **runmqslr**, to run the channel listener process.

Communications Server has a tuning parameter called the **Receive_Allocate timeout** parameter that is set in the Transaction Program. The default value of this parameter is 3600 and this indicates that the listener will only remain active for 3600 seconds, that is, 1 hour. You can make your listener run for longer than this by increasing the value of the **Receive_Allocate timeout** parameter. You can also make it run ‘forever’ by specifying zero.
What next?

The SNA configuration task is complete. From the **File** pull-down, select **Save** and specify a file name under which to save your SNA configuration information, for example, NTCONFIG (\SF5800001/SF590000). When prompted, select this configuration as the default.

From the SNA Node Operations application, start the node by clicking the **Start node** button on the toolbar. Specify the file name of the configuration you just saved. (It should appear in the file-name box by default, because you identified it as your default configuration.) When the node startup is complete, ensure that your link to the remote node has been established by selecting the **Connections** button on the toolbar, then find the link name you configured (for example, LINK0000). The link should be active if the remote node is active waiting for the link to be established.

A complementary SNA setup process is required on the node to which you are connecting before you can attempt WebSphere MQ server-to-server message transmissions.

The LU 6.2 connection is now established. You are ready to complete the configuration. Go to "WebSphere MQ for Windows configuration" on page 183.
Establishing a TCP connection

The TCP stack that is shipped with Windows systems do not include an inet
daemon or equivalent.

The WebSphere MQ command used to start a TCP listener is:
runmqslr -t tcp

The listener must be started explicitly before any channels are started.

What next?

When the TCP/IP connection is established, you are ready to complete the
configuration. Go to “WebSphere MQ for Windows configuration” on page 183.

Establishing a NetBIOS connection

A NetBIOS connection is initiated from a queue manager that uses the
ConnectionName parameter on its channel definition to connect to a target listener.
To set up a NetBIOS connection, follow these steps:

1. At each end of the channel specify the local NetBIOS name to be used by the
WebSphere MQ channel processes, in the Windows registry or in the queue
manager configuration file qm.ini. For example, the NETBIOS stanza in the
Windows registry at the sending end might look like this:

   NETBIOS:
   LocalName=WNTNETB1

   and at the receiving end:

   NETBIOS:
   LocalName=WNTNETB2

   Each WebSphere MQ process must use a different local NetBIOS name. Do not
use your machine name as the NetBIOS name because Windows already uses
it.

2. At each end of the channel, verify the LAN adapter number being used on
your system. The WebSphere MQ for Windows default for logical adapter
number 0 is NetBIOS running over a TCP/IP network. To use native NetBIOS
you need to select logical adapter number 1. See “Establishing the LAN adapter
number” on page 136

   Specify the correct LAN adapter number in the NETBIOS stanza of the the
Windows registry. For example:

   NETBIOS:
   AdapterNum=1

3. So that sender channel initiation will work, specify the local NetBIOS name via
the MQNAME environment variable:

   SET MQNAME=WNTNETB1I

   This name must be unique.

4. At the sending end, define a channel specifying the NetBIOS name being used
at the other end of the channel. For example:
DEFINE CHANNEL (WINNT.OS2.NET) CHLTYPE(SDR) +
    TRPTYPE(NETBIOS) +
    CONNAME(WNTNETB2) +
    XMITQ(OS2) +
    MCATYPE(THREAD) +
    REPLACE

You must specify the option MCATYPE(THREAD) because, on Windows, sender channels must be run as threads.

5. At the receiving end, define the corresponding receiver channel. For example:

    DEFINE CHANNEL (WINNT.OS2.NET) CHLTYPE(RCVR) +
        TRPTYPE(NETBIOS) +
        REPLACE

6. Start the channel initiator because each new channel is started as a thread rather than as a new process.

   runmqchi

7. At the receiving end, start the WebSphere MQ listener:

   runmqlsr -t netbios

Optionally you may specify values for the queue manager name, NetBIOS local name, number of sessions, number of names, and number of commands. See “Defining a NetBIOS connection” on page 135 for more information about setting up NetBIOS connections.

## Establishing an SPX connection

This section discusses the following topics:

- IPX/SPX parameters
- SPX addressing
- Receiving on SPX

### IPX/SPX parameters

Please refer to the Microsoft documentation for full details of the use and setting of the NWLink IPX and SPX parameters. The IPX/SPX parameters are in the following paths in the registry:

- HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Service\NWLinkSPX\Parameters
- HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Service\NWLinkIPX\Parameters
**SPX addressing**

WebSphere MQ uses the SPX address of each machine to establish connectivity. The SPX address is specified in the following form:

\[ \text{network.node(socket)} \]

Where:

- **network** is the 4-byte network address of the network on which the remote machine resides,
- **node** is the 6-byte node address, which is the LAN address of the LAN adapter in the remote machine,
- **socket** is the 2-byte socket number on which the remote machine will listen.

The default socket number used by WebSphere MQ is 5E86. You can change the default socket number by specifying it in the Windows NT registry or in the queue manager configuration file qm.ini. If you have taken the default options for installation, the qm.ini file for queue manager OS2 is found in `c:\mqm\qmgs\os2`. The lines in the Windows NT registry might read:

```ini
SPX:
  SOCKET=n
```

For more information about values you can set in qm.ini, see Appendix D, "Configuration file stanzas for distributed queuing" on page 715.

The SPX address is later specified in the CONNAME parameter of the sender channel definition. If the WebSphere MQ systems being connected reside on the same network, the network address need not be specified. Similarly, if the remote system is listening on the default socket number (5E86), it need not be specified. A fully qualified SPX address in the CONNAME parameter would be:

\[ \text{CONNAME('network.node(socket)' )} \]

but if the systems reside on the same network and the default socket number is used, the parameter would be:

\[ \text{CONNAME(node)} \]

A detailed example of the channel configuration parameters is given in "WebSphere MQ for Windows configuration" on page 183.

**Receiving on SPX**

Receiving channel programs are started in response to a startup request from the sending channel. To do this, a listener program has to be started to detect incoming network requests and start the associated channel.

You should use the WebSphere MQ listener.

**Using the WebSphere MQ listener**

To run the Listener supplied with WebSphere MQ, that starts new channels as threads, use the RUNMQLSR command. For example:

```cmd
RUNMQLSR -t spx
```

Optionally you may specify the queue manager name or the socket number if you are not using the defaults.
WebSphere MQ for Windows configuration

Notes:

1. You can use the sample program, AMQSBCG, to display the contents and headers of all the messages in a queue. For example:

   AMQSBCG q_name qmgr_name

   displays the contents of the queue q_name defined in queue manager qmgr_name.

   Alternatively, you can use the message browser in the WebSphere MQ Explorer.

2. The WebSphere MQ command used to start the TCP/IP listener is:

   runmqslr -t tcp

   The listener enables receiver channels to start automatically in response to a start request from an inbound sender channel.

3. You can start any channel from the command prompt using the command

   runmqch1 -c channel.name

4. Error logs can be found in the directories \mqm\qmgrs\qmgrname\errors and \mqm\qmgrs\@system\errors. In both cases, the most recent messages are at the end of amqerr01.log.

5. When you are using the command interpreter runmqsc to enter administration commands, a + at the end of a line indicates that the next line is a continuation. Ensure that there is a space between the last parameter and the continuation character.

Default configuration

You can create a default configuration by using either the First Steps application or the WebSphere MQ Postcard application to guide you through the process. For information about this, see the WebSphere MQ System Administration Guide book.

Basic configuration

You can create and start a queue manager from the WebSphere MQ Explorer or from the command prompt.

If you choose the command prompt:

1. Create the queue manager using the command:

   crtmqm -u dlqname -q winnt

   where:

   winnt Is the name of the queue manager
   -q Indicates that this is to become the default queue manager
   -u dlqname Specifies the name of the undeliverable message queue

   This command creates a queue manager and a set of default objects.

2. Start the queue manager using the command:

   strmqm winnt

   where winnt is the name given to the queue manager when it was created.
**Windows configuration**

**Channel configuration**

The following sections detail the configuration to be performed on the Windows queue manager to implement the channel described in Figure 32 on page 101.

In each case the MQSC command is shown. Either start `runmqsc` from a command prompt and enter each command in turn, or build the commands into a command file.

Examples are given for connecting WebSphere MQ for Windows and MQSeries for OS/2 Warp. If you wish to connect to another WebSphere MQ product use the appropriate set of values from the table in place of those for OS/2.

**Note:** The words in **bold** are user-specified and reflect the names of WebSphere MQ objects used throughout these examples. If you change the names used here, ensure that you also change the other references made to these objects throughout this book. All others are keywords and should be entered as shown.

**Table 18. Configuration worksheet for WebSphere MQ for Windows**

<table>
<thead>
<tr>
<th>Parameter Name</th>
<th>Reference</th>
<th>Example Used</th>
<th>User Value</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Definition for local node</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>A</strong> Queue Manager Name</td>
<td>WINNT</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>B</strong> Local queue name</td>
<td>WINNT.LOCALQ</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Connection to MQSeries for OS/2 Warp</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>The values in this section of the table must match those used in Table 16 on page 161 as indicated.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>C</strong> Remote queue manager name</td>
<td>OS2</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>D</strong> Remote queue name</td>
<td>OS2.REMOTEQ</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>E</strong> Queue name at remote system</td>
<td>OS2.LOCALQ</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>F</strong> Transmission queue name</td>
<td>OS2</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>G</strong> Sender (SNA) channel name</td>
<td>WINNT.OS2.SNA</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>H</strong> Sender (TCP/IP) channel name</td>
<td>WINNT.OS2.TCP</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>I</strong> Receiver (SNA) channel name</td>
<td>OS2.WINNT.SNA</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>J</strong> Receiver (TCP) channel name</td>
<td>OS2.WINNT.TCP</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>K</strong> Receiver (NetBIOS) channel name</td>
<td>WINNT.OS2.NET</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>L</strong> Receiver (SPX) channel name</td>
<td>WINNT.OS2.SPX</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>M</strong> Receiver (NetBIOS) channel name</td>
<td>OS2.WINNT.NET</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>N</strong> Receiver (SPX) channel name</td>
<td>OS2.WINNT.SPX</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Connection to WebSphere MQ for AIX</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>The values in this section of the table must match those used in Table 22 on page 210 as indicated.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>C</strong> Remote queue manager name</td>
<td>AIX</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>D</strong> Remote queue name</td>
<td>AIX.REMOTEQ</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>E</strong> Queue name at remote system</td>
<td>AIX.LOCALQ</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>F</strong> Transmission queue name</td>
<td>AIX</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>G</strong> Sender (SNA) channel name</td>
<td>WINNT.AIX.SNA</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>H</strong> Sender (TCP) channel name</td>
<td>WINNT.AIX.TCP</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>I</strong> Receiver (SNA) channel name</td>
<td>AIX.WINNT.SNA</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>J</strong> Receiver (TCP) channel name</td>
<td>AIX.WINNT.TCP</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Connection to MQSeries for Compaq Tru64 UNIX</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>The values in this section of the table must match those used in Table 23 on page 216 as indicated.</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

184  WebSphere MQ Intercommunication
### Table 18. Configuration worksheet for WebSphere MQ for Windows (continued)

<table>
<thead>
<tr>
<th>Parameter Name</th>
<th>Reference</th>
<th>Example Used</th>
<th>User Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>C Remote queue manager name</td>
<td>A</td>
<td>DECUX</td>
<td></td>
</tr>
<tr>
<td>D Remote queue name</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>E Queue name at remote system</td>
<td>B</td>
<td>DECUX.LOCALQ</td>
<td></td>
</tr>
<tr>
<td>F Transmission queue name</td>
<td></td>
<td>DECUX</td>
<td></td>
</tr>
<tr>
<td>H Sender (TCP) channel name</td>
<td></td>
<td>DECUX.WINNT.TCP</td>
<td>WINNT.DECUX.TCP</td>
</tr>
<tr>
<td>J Receiver (TCP) channel name</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Connection to WebSphere MQ for HP-UX**

The values in this section of the table must match those used in Table 25 on page 239, as indicated.

| C Remote queue manager name | A         | HPUX           |                          |
| D Remote queue name         |           |                |                          |
| E Queue name at remote system | B       | HPUX.LOCALQ    |                          |
| F Transmission queue name   |           | HPUX           |                          |
| G Sender (SNA) channel name |           | WINNT.HPUX.SNA | WINNT.HPUX.TCP           |
| I Receiver (SNA) channel name |         |                |                          |
| J Receiver (TCP/IP) channel name |        |                |                          |

**Connection to MQSeries for AT&T GIS UNIX**

The values in this section of the table must match those used in Table 27 on page 253, as indicated.

| C Remote queue manager name | A         | GIS            |                          |
| D Remote queue name         |           |                |                          |
| E Queue name at remote system | B       | GIS.LOCALQ     |                          |
| F Transmission queue name   |           | GIS            |                          |
| G Sender (SNA) channel name |           | WINNT.GIS.SNA  | WINNT.GIS.TCP            |
| I Receiver (SNA) channel name |         |                |                          |
| J Receiver (TCP/IP) channel name |        |                |                          |

**Connection to WebSphere MQ for Solaris**

The values in this section of the table must match those used in Table 30 on page 290, as indicated.

| C Remote queue manager name | A         | SOLARIS        |                          |
| D Remote queue name         |           |                |                          |
| E Queue name at remote system | B       | SOLARIS.LOCALQ |                          |
| F Transmission queue name   |           | SOLARIS        |                          |
| G Sender (SNA) channel name |           | WINNT.SOLARIS.SNA | WINNT.SOLARIS.TCP     |
| I Receiver (SNA) channel name |         |                |                          |
| J Receiver (TCP) channel name |        |                |                          |

**Connection to WebSphere MQ for iSeries**

The values in this section of the table must match those used in Table 49 on page 535, as indicated.

| C Remote queue manager name | A         | AS400          |                          |
| D Remote queue name         |           |                |                          |
| E Queue name at remote system | B       | AS400.LOCALQ   |                          |
| F Transmission queue name   |           | AS400          |                          |
| G Sender (SNA) channel name |           | WINNT.AS400.SNA | WINNT.AS400.TCP         |
| H Sender (TCP) channel name |           |                |                          |
Table 18. Configuration worksheet for WebSphere MQ for Windows (continued)

<table>
<thead>
<tr>
<th>Parameter Name</th>
<th>Reference</th>
<th>Example Used</th>
<th>User Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Receiver (SNA) channel name</td>
<td>/SF580000I/SF590000</td>
<td>AS400.WINNT.SNA</td>
<td></td>
</tr>
<tr>
<td>Receiver (TCP) channel name</td>
<td>/SF580000G/SF590000</td>
<td>AS400.WINNT.TCP</td>
<td></td>
</tr>
</tbody>
</table>

**Connection to WebSphere MQ for z/OS without CICS**

The values in this section of the table must match those used in [Table 34 on page 372](#) as indicated.

<table>
<thead>
<tr>
<th>Parameter Name</th>
<th>Reference</th>
<th>Example Used</th>
<th>User Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Remote queue manager name</td>
<td>/SF580000C/SF590000</td>
<td>MVS</td>
<td></td>
</tr>
<tr>
<td>Remote queue name</td>
<td>/SF580000D/SF590000</td>
<td>MVS.REMOTEQ</td>
<td></td>
</tr>
<tr>
<td>Queue name at remote system</td>
<td>/SF580000E/SF590000</td>
<td>MVS.LOCALQ</td>
<td></td>
</tr>
<tr>
<td>Transmission queue name</td>
<td>/SF580000F/SF590000</td>
<td>MVS</td>
<td></td>
</tr>
<tr>
<td>Sender (SNA) channel name</td>
<td>/SF580000G/SF590000</td>
<td>WINNT.MVS.SNA</td>
<td></td>
</tr>
<tr>
<td>Sender (TCP) channel name</td>
<td>/SF580000H/SF590000</td>
<td>WINNT.MVS.TCP</td>
<td></td>
</tr>
<tr>
<td>Receiver (SNA) channel name</td>
<td>/SF580000I/SF590000</td>
<td>MVS.WINNT.SNA</td>
<td></td>
</tr>
<tr>
<td>Receiver (TCP/IP) channel name</td>
<td>/SF580000J/SF590000</td>
<td>MVS.WINNT.TCP</td>
<td></td>
</tr>
</tbody>
</table>

**Connection to WebSphere MQ for z/OS using queue-sharing groups**

The values in this section of the table must match those used in [Table 43 on page 454](#) as indicated.

<table>
<thead>
<tr>
<th>Parameter Name</th>
<th>Reference</th>
<th>Example Used</th>
<th>User Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Remote queue manager name</td>
<td>/SF580000C/SF590000</td>
<td>QSG</td>
<td></td>
</tr>
<tr>
<td>Remote queue name</td>
<td>/SF580000D/SF590000</td>
<td>QSG.REMOTEQ</td>
<td></td>
</tr>
<tr>
<td>Queue name at remote system</td>
<td>/SF580000E/SF590000</td>
<td>QSG.SHAREDQ</td>
<td></td>
</tr>
<tr>
<td>Transmission queue name</td>
<td>/SF580000F/SF590000</td>
<td>QSG</td>
<td></td>
</tr>
<tr>
<td>Sender (SNA) channel name</td>
<td>/SF580000G/SF590000</td>
<td>WINNT.QSG.SNA</td>
<td></td>
</tr>
<tr>
<td>Sender (TCP) channel name</td>
<td>/SF580000H/SF590000</td>
<td>WINNT.QSG.TCP</td>
<td></td>
</tr>
<tr>
<td>Receiver (SNA) channel name</td>
<td>/SF580000I/SF590000</td>
<td>QSG.WINNT.SNA</td>
<td></td>
</tr>
<tr>
<td>Receiver (TCP/IP) channel name</td>
<td>/SF580000J/SF590000</td>
<td>QSG.WINNT.TCP</td>
<td></td>
</tr>
</tbody>
</table>

**Connection to MQSeries for VSE/ESA**

The values in this section of the table must match those used in [Table 51 on page 554](#) as indicated.

<table>
<thead>
<tr>
<th>Parameter Name</th>
<th>Reference</th>
<th>Example Used</th>
<th>User Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Remote queue manager name</td>
<td>/SF580000C/SF590000</td>
<td>VSE</td>
<td></td>
</tr>
<tr>
<td>Remote queue name</td>
<td>/SF580000D/SF590000</td>
<td>VSE.REMOTEQ</td>
<td></td>
</tr>
<tr>
<td>Queue name at remote system</td>
<td>/SF580000E/SF590000</td>
<td>VSE.LOCALQ</td>
<td></td>
</tr>
<tr>
<td>Transmission queue name</td>
<td>/SF580000F/SF590000</td>
<td>VSE</td>
<td></td>
</tr>
<tr>
<td>Sender channel name</td>
<td>/SF580000G/SF590000</td>
<td>WINNT.VSE.SNA</td>
<td></td>
</tr>
<tr>
<td>Receiver channel name</td>
<td>/SF580000I/SF590000</td>
<td>VSE.WINNT.SNA</td>
<td></td>
</tr>
</tbody>
</table>

### WebSphere MQ for Windows sender-channel definitions using SNA

- def q1 (OS2) +
  - usage(xmitq) +
  - replace

- def qr (OS2.REMOTEQ) +
  - rname(OS2.LOCALQ) +
  - rqmname(OS2) +
  - xmitq(OS2) +
  - replace

- def chl (WINNT.OS2.SNA) chlname(sdr) +
  - trtype(lu62) +
  - conname(OS2CPIC) +
  - xmitq(OS2) +
  - replace

---

186  WebSphere MQ Intercommunication
WebSphere MQ for Windows receiver-channel definitions using SNA

def ql (WINNT.LOCALQ) replace

def chl (OS2.WINNT.SNA) chltype(rcvr) +
    trptype(lu62) +
    replace

WebSphere MQ for Windows sender-channel definitions using TCP/IP

def ql (OS2) +
    usage(xmitq) +
    replace

def qr (OS2.REMOTEQ) +
    rname(OS2.LOCALQ) +
    rqmname(OS2) +
    xmitq(OS2) +
    replace

def chl (WINNT.OS2.TCP) chltype(sdr) +
    trptype(tcp) +
    conname(remote_tcpip_hostname) +
    xmitq(OS2) +
    replace

WebSphere MQ for Windows receiver-channel definitions using TCP

def ql (WINNT.LOCALQ) replace

def chl (OS2.WINNT.TCP) chltype(rcvr) +
    trptype(tcp) +
    replace

WebSphere MQ for Windows sender-channel definitions using NetBIOS

def ql (OS2) +
    usage(xmitq) +
    replace

def qr (OS2.REMOTEQ) +
    rname(OS2.LOCALQ) +
    rqmname(OS2) +
    xmitq(OS2) +
    replace

def chl (WINNT.OS2.NET) chltype(sdr) +
    trptype(netbios) +
    conname(remote_system_NetBIOS_name) +
    xmitq(OS2) +
    replace

WebSphere MQ for Windows receiver-channel definitions using NetBIOS

def ql (WINNT.LOCALQ) replace

def chl (OS2.WINNT.NET) chltype(rcvr) +
    trptype(tcp) +
    replace
WebSphere MQ for Windows sender-channel definitions using SPX

def q1 (OS2) +
   usage(xmitq) +
   replace

def qr (OS2.REMOTEQ) +
   rname(OS2.LOCALQ) +
   rqmname(OS2) +
   xmitq(OS2) +
   replace

def chl (WINNT.OS2.SPX) chltype(sdr) +
   trptype(spx) +
   conname('network.node(socket)') +
   xmitq(OS2) +
   replace

WebSphere MQ for Windows receiver-channel definitions using SPX

def q1 (WINNT.LOCALQ) replace

def chl (OS2.WINNT.SPX) chltype(rcvr) +
   trptype(tcp) +
   replace

Automatic startup

WebSphere MQ for Windows allows you to automate the startup of a queue manager and its channel initiator, channels, listeners, and command servers. Use the IBM WebSphere MQ Services snap-in to define the services for the queue manager. When you have successfully completed testing of your communications setup, set the relevant services to automatic within the snap-in. This file can be read by the supplied WebSphere MQ service when the system is started.

For more information about this, see the WebSphere MQ System Administration Guide book.

Running channels as processes or threads

WebSphere MQ for Windows provides the flexibility to run sender channels as Windows NT processes or Windows NT threads. This is specified in the MCATYPE parameter on the sender channel definition. Each installation should select the type appropriate for their application and configuration. Factors affecting this choice are discussed below.

Most installations will select to run their sender channels as threads, because the virtual and real memory required to support a large number of concurrent channel connections will be reduced. When the WebSphere MQ listener process (started via the RUNMQLSR command) exhausts the available private memory needed, an additional listener process will need to be started to support more channel connections. When each channel runs as a process, additional processes are automatically started, avoiding the out-of-memory condition.

If all channels are run as threads under one WebSphere MQ listener, a failure of the listener for any reason will cause all channel connections to be temporarily lost. This can be prevented by balancing the threaded channel connections across two or more listener processes, thus enabling other connections to keep running. If each sender channel is run as a separate process, the failure of the listener for that process will affect only that specific channel connection.
A NetBIOS connection needs a separate process for the Message Channel Agent. Therefore, before you can issue a START CHANNEL command, you must start the channel initiator, or you may start a channel using the RUNMQCHL command.

**Multiple thread support — pipelining**

You can optionally allow a message channel agent (MCA) to transfer messages using multiple threads. This process, called pipelining, enables the MCA to transfer messages more efficiently, with fewer wait states, which improves channel performance. Each MCA is limited to a maximum of two threads.

You control pipelining with the `PipeLineLength` parameter in the qm.ini file. This parameter is added to the CHANNELS stanza:

```
PipeLineLength=1\number
```

This attribute specifies the maximum number of concurrent threads a channel will use. The default is 1. Any value greater than 1 will be treated as 2.

With WebSphere MQ for Windows, use the WebSphere MQ Services snap-in to set the `PipeLineLength` parameter in the registry. Refer to the *WebSphere MQ System Administration Guide* book for a complete description of the CHANNELS stanza.

**Notes:**

1. `PipeLineLength` applies only to V5.2 or later products.
2. Pipelining is effective only for TCP/IP channels.

When you use pipelining, the queue managers at both ends of the channel must be configured to have a `PipeLineLength` greater than 1.

**Channel exit considerations**

Note that pipelining can cause some exit programs to fail, because:

- Exits might not be called serially.
- Exits might be called alternately from different threads.
Check the design of your exit programs before you use pipelining:

- Exits must be reentrant at all stages of their execution.
- When you use MQI calls, remember that MQI handles are thread-specific.

Consider a message exit that opens a queue and uses its handle for MQPUT calls on all subsequent invocations of the exit. This fails in pipelining mode because the exit is called from different threads. To avoid this failure, keep a queue handle for each thread and check the thread identifier each time the exit is invoked.
Chapter 13. Setting up communication in UNIX systems

DQM is a remote queuing facility for WebSphere MQ. It provides channel control programs for the queue manager which form the interface to communication links, controllable by the system operator. The channel definitions held by distributed-queuing management use these connections.

When a distributed-queuing management channel is started, it tries to use the connection specified in the channel definition. For this to succeed, it is necessary for the connection to be defined and available. This chapter explains how to do this. You might also find it helpful to refer to the following chapters:

- Chapter 14, “Example configuration - IBM WebSphere MQ for AIX” on page 197
- Chapter 15, “Example configuration - IBM MQSeries for Compaq Tru64 UNIX” on page 215
- Chapter 16, “Example configuration - IBM WebSphere MQ for HP-UX” on page 219
- Chapter 17, “Example configuration - IBM MQSeries for AT&T GIS UNIX, V2.2” on page 243
- Chapter 18, “Example configuration - IBM WebSphere MQ for Solaris” on page 257

For OS/2 and Windows, see Chapter 10, “Setting up communication for OS/2 and Windows” on page 129. For Compaq OpenVMS Alpha, see Chapter 19, “Setting up communication in Compaq OpenVMS Alpha systems” on page 295. For Compaq NonStop Kernel, see Chapter 20, “Setting up communication in Compaq NonStop Kernel” on page 307.

Deciding on a connection

There are three forms of communication for WebSphere MQ on UNIX systems:
- TCP
- LU 6.2
- UDP (WebSphere MQ for AIX only)

Each channel definition must specify one only as the transmission protocol (Transport Type) attribute. One or more protocols may be used by a queue manager.

For WebSphere MQ clients, it may be useful to have alternative channels using different transmission protocols. See the WebSphere MQ Clients book.

Defining a TCP connection

The channel definition at the sending end specifies the address of the target. The inetd daemon is configured for the connection at the receiving end.

Sending end

Specify the host name, or the TCP address of the target machine, in the Connection Name field of the channel definition. The port to connect to will default to 1414. Port number 1414 is assigned by the Internet Assigned Numbers Authority to WebSphere MQ.
Defining a TCP connection

To use a port number other than the default, change the connection name field thus:

Connection Name REMHOST(1822)

where REMHOST is the hostname of the remote machine and 1822 is the port number required. (This must be the port that the listener at the receiving end is listening on.)

Alternatively you can change the port number by specifying it in the queue manager configuration file (qm.ini):

TCP:
   Port=1822

For more information about the values you set using QM.INI, see Appendix D, “Configuration file stanzas for distributed queuing” on page 715.

Receiving on TCP

You should use either the TCP/IP listener (INETD) or the WebSphere MQ listener.

Using the TCP/IP listener

To start channels on UNIX, the /etc/services file and the inetd.conf file must be edited, following the instructions below:

1. Edit the /etc/services file:

   Note: To edit the /etc/services file, you must be logged in as a superuser or root. You can change this, but it must match the port number specified at the sending end.

   Add the following line to the file:

   MQSeries 1414/tcp

   where 1414 is the port number required by WebSphere MQ.

2. Add a line in the inetd.conf file to call the program amqcrsta:

   MQSeries stream tcp nowait mqm /mgtop/bin/amqcrsta amqcrsta
   [-m Queue_Man_Name]

The updates are active after inetd has reread the configuration files. To do this, issue the following commands from the root user ID:

- On AIX:
  refresh -s inetd

- On other UNIX systems:
  kill -1 <process number>

When the listener program started by INETD inherits the locale from INETD, it is possible that the MQMDE will not be honored (merged) and will be placed on the queue as message data. To ensure that the MQMDE is honored, you must set the locale correctly. The locale set by INETD may not match that chosen for other locales used by WebSphere MQ processes. To set the locale:

1. Create a shell script which sets the locale environment variables LANG, LC_COLLATE, LC_CTYPE, LC_MONETARY, LC_NUMERIC, LC_TIME, and LC_MESSAGES to the locale used for other WebSphere MQ process.

2. In the same shell script, call the listener program.

3. Modify the inetd.conf file to call your shell script in place of the listener program.
Defining a TCP connection

It is possible to have more than one queue manager on the server machine. You must add a line to each of the two files, as above, for each of the queue managers. For example:

MQSeries1 1414/tcp
MQSeries2 1822/tcp
MQSeries2 stream tcp nowait mqm /mqmtop/bin/amqcrsta amqcrsta -m QM2

This avoids error messages being generated if there is a limitation on the number of outstanding connection requests queued at a single TCP port. For information about the number of outstanding connection requests, see "Using the TCP listener backlog option."

Using the TCP listener backlog option

When receiving on TCP, a maximum number of outstanding connection requests is set. This can be considered a backlog of requests waiting on the TCP port for the listener to accept the request. The default listener backlog values are shown in Table 19.

Table 19. Default outstanding connection requests

<table>
<thead>
<tr>
<th>Platform</th>
<th>Default listener backlog value</th>
</tr>
</thead>
<tbody>
<tr>
<td>AIX V4.2 or later</td>
<td>100</td>
</tr>
<tr>
<td>AIX V4.1</td>
<td>10</td>
</tr>
<tr>
<td>HP-UX</td>
<td>20</td>
</tr>
<tr>
<td>Solaris</td>
<td>100</td>
</tr>
<tr>
<td>Linux</td>
<td>100</td>
</tr>
<tr>
<td>All others</td>
<td>5</td>
</tr>
</tbody>
</table>

If the backlog reaches the values shown in Table 19, the TCP/IP connection is rejected and the channel will not be able to start.

For MCA channels, this results in the channel going into a RETRY state and retrying the connection at a later time.

For client connections, the client receives an MQRC_Q_MGR_NOT_AVAILABLE reason code from MQCONN and should retry the connection at a later time.

However, to avoid this error, you can add an entry in the qm.ini file:

TCP:
ListenerBacklog = n

This overrides the default maximum number of outstanding requests (see Table 19) for the TCP/IP listener.

Note: Some operating systems support a larger value than the default. If necessary, this can be used to avoid reaching the connection limit.

To run the listener with the backlog option switched on, use the RUNMQLSR -B command. For information about the RUNMQLSR command, see the WebSphere MQ System Administration Guide book.

Using the WebSphere MQ listener

To run the listener supplied with WebSphere MQ, which starts new channels as threads, use the runmqlsr command. For example:
Defining a TCP connection

runmqlsr -t tcp [-m QMNAME] [-p 1822]

The square brackets indicate optional parameters; QMNAME is not required for the default queue manager, and the port number is not required if you are using the default (1414).

For the best performance, run the WebSphere MQ listener as a trusted application as described in “Running channels and listeners as trusted applications” on page 126. See the WebSphere MQ Application Programming Guide for information about trusted applications.

You can stop all WebSphere MQ listeners running on a queue manager that is inactive, using the command:

endmqslr [-m QMNAME]

If you do not specify a queue manager name, the default queue manager is assumed.

Using the TCP/IP SO_KEEPALIVE option

If you want to use the SO_KEEPALIVE option (as discussed in “Checking that the other end of the channel is still available” on page 66) you must add the following entry to your queue manager configuration file (QM.INI) or the Windows NT registry:

TCP:
KeepAlive=yes

On some UNIX systems, you can define how long TCP waits before checking that the connection is still available, and how frequently it retries the connection if the first check fails. This is either a kernel tunable parameter, or can be entered at the command line. See the documentation for your UNIX system for more information.

On MQSeries for S/390 and DC/OSx you can set the TCP keepalive parameters by using the idtune and idbuild commands to modify the TCP_KEEPCNT and TCP_KEEPINT values for the kernel configuration. The default configuration is to retry 7 times at 7200 second (2 hourly) intervals.

Defining an LU 6.2 connection

SNA must be configured so that an LU 6.2 conversation can be established between the two machines.

See the Multiplatform APPC Configuration Guide and the following table for information.

<table>
<thead>
<tr>
<th>Remote platform</th>
<th>TPNAME</th>
<th>TPPATH</th>
</tr>
</thead>
<tbody>
<tr>
<td>z/OS or OS/390</td>
<td>The same as the corresponding TPName in the side information on the remote queue manager.</td>
<td>-</td>
</tr>
<tr>
<td>or MVS/ESA without CICS</td>
<td>CKRC (sender) CKSV (requester)</td>
<td>CKRC (server)</td>
</tr>
<tr>
<td>z/OS or OS/390</td>
<td>The same as the compare value in the routing entry on the OS/400 system.</td>
<td>-</td>
</tr>
<tr>
<td>or MVS/ESA using CICS</td>
<td></td>
<td></td>
</tr>
<tr>
<td>OS/400</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Defining an LU 6.2 connection

#### Table 20. Settings on the local UNIX system for a remote queue manager platform (continued)

<table>
<thead>
<tr>
<th>Remote platform</th>
<th>TPNAME</th>
<th>TPATH</th>
</tr>
</thead>
<tbody>
<tr>
<td>OS/2</td>
<td>As specified in the OS/2 Run Listener command, or defaulted from the OS/2 queue manager configuration file.</td>
<td>&lt;drive&gt;:\mqm\bin\amqcrs6a</td>
</tr>
<tr>
<td>UNIX systems</td>
<td>The same as the corresponding TPName in the side information on the remote queue manager.</td>
<td>mqmtop/bin/amqcrs6a</td>
</tr>
<tr>
<td>Windows</td>
<td>As specified in the Windows Run Listener command, or the invokable Transaction Program that was defined using TpSetup on Windows.</td>
<td>&lt;drive&gt;:\mqm\bin\amqcrs6a</td>
</tr>
</tbody>
</table>

If you have more than one queue manager on the same machine, ensure that the TPnames in the channel definitions are unique.

#### Sending end

- On UNIX systems other than SINIX, and DC/OSx, create a CPI-C side object (symbolic destination) and enter this name in the Connection name field in the channel definition. Also create an LU 6.2 link to the partner.

In the CPI-C side object enter the partner LU name at the receiving machine, the transaction program name and the mode name. For example:

Partner LU Name: REMHOST
Remote TP Name: recv
Service Transaction Program: no
Mode Name: #INTER

On HP-UX, use the APPCLLU environment variable to name the local LU that the sender should use. On Solaris, set the APPC_LOCAL_LU environment variable to be the local LU name.

SECURITY PROGRAM is used, where supported by CPI-C, when WebSphere MQ attempts to establish an SNA session.

- On SINIX, create an XSYMDEST entry in SNA configuration file (the TRANSIT KOGS file), for example:

```plaintext
XSYMDEST sendMP01,
  RLU = forties,
  MODE = MODE1,
  TP = recvMP01,
  TP-TYP = USER,
  SEC-TYP = NONE
```

See the MQSeries for SINIX and DC/OSx System Management Guide for more information about the TRANSIT KOGS file.

- On DC/OSx, create an entry in the /etc/opt/lu62/cpic_cfg file, for example:

```plaintext
sendMP01 <local LU name> <remote LU name> <mode name> <remote TP name>
```

#### Receiving on LU 6.2

- On UNIX systems other than SINIX, and DC/OSx, create a listening attachment at the receiving end, an LU 6.2 logical connection profile, and a TPN profile.
In the TPN profile, enter the full path to the executable and the Transaction Program name:

**Full path to TPN executable**: `mqmtop/bin/amqcrs6a`

**Transaction Program name**: `recv`

**User ID**: `0`

On systems where you can set the User ID, you should specify a user who is a member of the `mqm` group. On HP-UX, set the `APPCTPN` (transaction name) and `APPCLLU` (local LU name) environment variables (you can use the configuration panels for the invoked transaction program). On Solaris, set the `APPC_LOCAL_LU` environment variable to be the local LU name.

On Solaris, `amqcrs6a` requires the option `-n tp_name`, where `tp_name` is the TP name on the receiving end of the SNA connection. It is the value of the `tp_path` variable in the SunLink configuration file.

You may need to use a queue manager other than the default queue manager. If so, define a command file that calls:

`amqcrs6a -m Queue_Man_Name`

then call the command file. On AIX, use the TPN profile parameters as follows:

- **Use Command Line Parameters**: yes
- **Command Line Parameters**: `-m Queue_Man_Name`

- On SINE, create an XTP entry in the SNA configuration file (the TRANSIT KOGS file), for example:

  ```
  XTP recvMP01,
  UID = abcdefgh,
  TYP = USER,
  PATH = /home/abcdefgh/recvMP01.sh,
  SECURE = NO
  ```

  Where `/home/abcdefgh/recvMP01.sh` is a file that contains:

  ```
  #!/bin/sh
  # script to start the receiving side for the qmgr MP01
  exec /opt/mqm/bin/amqcrs6a -m <queue manager>
  ```

  See the *MQSeries for SINE and DC/OSx System Management Guide* for more information about the TRANSIT KOGS file.

- On DC/OSx, add a Transaction Program entry to the SNA configuration file, including the following information:

  ```
  TRANSACTION PROGRAM
  transaction program name (ebcdic): recvMP04
  transaction program execute name: '/home/abcdefgh/recvMP04.sh'
  tp is enabled
  tp supports basic conversations
  tp supports mapped conversations
  tp supports confirm synchronization
  tp supports no synchronization
  no verification is required
  number of pip fields required: 0
  privilege mask (hex): 0 (no privileges)
  ```
Chapter 14. Example configuration - IBM WebSphere MQ for AIX

This chapter gives an example of how to set up communication links from WebSphere MQ for AIX to WebSphere MQ products on the following platforms:
- OS/2
- Windows
- Compaq Tru64 UNIX
- HP-UX
- AT&T GIS UNIX³
- Solaris
- OS/400
- z/OS without CICS
- VSE/ESA

First it describes the parameters needed for an LU 6.2 connection, then it describes “Establishing a TCP connection” on page 208 and “Establishing a UDP connection” on page 209.

Once the connection is established, you need to define some channels to complete the configuration. This is described in “WebSphere MQ for AIX configuration” on page 209.

See Chapter 7, “Example configuration chapters in this book” on page 101 for background information about this chapter and how to use it.

Configuration parameters for an LU 6.2 connection

Table 21 presents a worksheet listing all the parameters needed to set up communication from AIX to one of the other WebSphere MQ platforms. The worksheet shows examples of the parameters, which have been tested in a working environment, and leaves space for you to fill in your own values. An explanation of the parameter names follows the worksheet. Use the worksheet in this chapter in conjunction with the worksheet in the chapter for the platform to which you are connecting.

Configuration worksheet

Use the following worksheet to record the values you will use for this configuration. Where numbers appear in the Reference column they indicate that the value must match that in the appropriate worksheet elsewhere in this book. The examples that follow in this chapter refer back to the values in the ID column of this table. The entries in the Parameter Name column are explained in “Explanation of terms” on page 200.

Table 21. Configuration worksheet for Communications Server for AIX

<table>
<thead>
<tr>
<th>ID</th>
<th>Parameter Name</th>
<th>Reference</th>
<th>Example</th>
<th>User Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Network name</td>
<td></td>
<td>NETID</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Control Point name</td>
<td></td>
<td>AIXPU</td>
<td></td>
</tr>
</tbody>
</table>

3. This platform has become NCR UNIX SVR4 MP-RAS, R3.0

© Copyright IBM Corp. 1993, 2002
### AIX and LU 6.2

**Table 21. Configuration worksheet for Communications Server for AIX (continued)**

<table>
<thead>
<tr>
<th>ID</th>
<th>Parameter Name</th>
<th>Reference</th>
<th>Example</th>
<th>User Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>Node ID</td>
<td></td>
<td>07123456</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Local LU name</td>
<td>AIXLU</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Local LU alias</td>
<td>AIXQMGR</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>TP Name</td>
<td>MQSERIES</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>Full path to TP executable</td>
<td></td>
<td><code>usr/lpp/mqm/bin/amqcrs6a</code></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>Token-ring adapter address</td>
<td></td>
<td><code>123456789012</code></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>Mode name</td>
<td>#INTER</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Connection to an OS/2 system**

The values in this section of the table must match those used in Table 15 on page 144, as indicated.

<table>
<thead>
<tr>
<th>ID</th>
<th>Parameter Name</th>
<th>Reference</th>
<th>Example</th>
<th>User Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>Network name</td>
<td>2 NETID</td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>Remote LU name</td>
<td>6 OS2LU</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>Remote Transaction Program name</td>
<td>8 MQSERIES</td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>LU 6.2 CPI-C Side Information profile name</td>
<td>OS2CPIC</td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>Mode name</td>
<td>17 #INTER</td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>LAN destination address</td>
<td>10 10005AFC5D83</td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>Token-Ring Link Station profile name</td>
<td>OS2PRO</td>
<td></td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>CP name of adjacent node</td>
<td>3 OS2PU</td>
<td></td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>LU 6.2 partner location profile name</td>
<td>OS2LOCPRO</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Connection to a Windows system**

The values in this section of the table must match those used in Table 17 on page 168, as indicated.

<table>
<thead>
<tr>
<th>ID</th>
<th>Parameter Name</th>
<th>Reference</th>
<th>Example</th>
<th>User Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>Network name</td>
<td>2 NETID</td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>Remote LU name</td>
<td>5 WINNTLU</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>Remote Transaction Program name</td>
<td>7 MQSERIES</td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>LU 6.2 CPI-C Side Information profile name</td>
<td>NTCPIC</td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>Mode name</td>
<td>17 #INTER</td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>LAN destination address</td>
<td>9 08005AA5FAB9</td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>Token-Ring Link Station profile name</td>
<td>NTNPRO</td>
<td></td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>CP name of adjacent node</td>
<td>3 WINNTPRO</td>
<td></td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>LU 6.2 partner LU profile name</td>
<td>NTLUPRO</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Connection to an HP-UX system**

The values in this section of the table must match those used in Table 24 on page 219, as indicated.

<table>
<thead>
<tr>
<th>ID</th>
<th>Parameter Name</th>
<th>Reference</th>
<th>Example</th>
<th>User Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>Network name</td>
<td>4 NETID</td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>Remote LU name</td>
<td>5 HPUXLU</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>Remote Transaction Program name</td>
<td>7 MQSERIES</td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>LU 6.2 CPI-C Side Information profile name</td>
<td>HPUXCPIC</td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>Mode name</td>
<td>6 #INTER</td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>LAN destination address</td>
<td>8 100090DC2C7C</td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>Token-Ring Link Station profile name</td>
<td>HPUXPRO</td>
<td></td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>CP name of adjacent node</td>
<td>2 HPUXPU</td>
<td></td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>LU 6.2 partner LU profile name</td>
<td>HPUXLU</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

---

198 WebSphere MQ Intercommunication
### Table 21. Configuration worksheet for Communications Server for AIX (continued)

<table>
<thead>
<tr>
<th>ID</th>
<th>Parameter Name</th>
<th>Reference</th>
<th>Example</th>
<th>User Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>Network name</td>
<td>2</td>
<td>NETID</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>Remote LU name</td>
<td>4</td>
<td>GISLU</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>Remote Transaction Program name</td>
<td>5</td>
<td>MQSERIES</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>LU 6.2 CPI-C Side Information profile name</td>
<td></td>
<td>GISCPI</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>Mode name</td>
<td>7</td>
<td>#INTER</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>LAN destination address</td>
<td>8</td>
<td>10007038E86B</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>Token-Ring Link Station profile name</td>
<td>9</td>
<td>GISPRO</td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>CP name of adjacent node</td>
<td>10</td>
<td>GISPU</td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>LU 6.2 partner LU profile name</td>
<td>11</td>
<td>GISLUPRO</td>
<td></td>
</tr>
</tbody>
</table>

**Connection to an AT&T GIS UNIX system**

The values in this section of the table must match those used in Table 26 on page 242 as indicated.

**Connection to a Solaris system**

The values in this section of the table must match those used in Table 28 on page 258 as indicated.

**Connection to an iSeries system**

The values in this section of the table must match those used in Table 48 on page 521 as indicated.

**Connection to a z/OS system without CICS**

The values in this section of the table must match those used in Table 33 on page 366 as indicated.
### AIX and LU 6.2

#### Table 21. Configuration worksheet for Communications Server for AIX (continued)

<table>
<thead>
<tr>
<th>ID</th>
<th>Parameter Name</th>
<th>Reference</th>
<th>Example</th>
<th>User Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>16</td>
<td>Token-Ring Link Station profile name</td>
<td></td>
<td>MVSPRO</td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>CP name of adjacent node</td>
<td>3</td>
<td>MVSPU</td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>LU 6.2 partner LU profile name</td>
<td></td>
<td>MVSLUPRO</td>
<td></td>
</tr>
</tbody>
</table>

#### Connection to a z/OS system using a generic interface

The values in this section of the table must match those used in Table 33 on page 366, as indicated.

<table>
<thead>
<tr>
<th>ID</th>
<th>Parameter Name</th>
<th>Reference</th>
<th>Example</th>
<th>User Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>Network name</td>
<td>2</td>
<td>NETID</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>Remote LU name</td>
<td>10</td>
<td>MYSGR</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>Remote Transaction Program name</td>
<td>7</td>
<td>MQSERIES</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>LU 6.2 CPI-C Side Information profile name</td>
<td></td>
<td>MVSCPIC</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>Mode name</td>
<td>6</td>
<td>#INTER</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>LAN destination address</td>
<td>8</td>
<td>400074511092</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>Token-Ring Link Station profile name</td>
<td>6</td>
<td>MVSPRO</td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>CP name of adjacent node</td>
<td>3</td>
<td>MVSPU</td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>LU 6.2 partner LU profile name</td>
<td></td>
<td>MVSLUPRO</td>
<td></td>
</tr>
</tbody>
</table>

#### Connection to a VSE/ESA system

The values in this section of the table must match those used in Table 50 on page 549, as indicated.

<table>
<thead>
<tr>
<th>ID</th>
<th>Parameter Name</th>
<th>Reference</th>
<th>Example</th>
<th>User Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>Network name</td>
<td>1</td>
<td>NETID</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>Remote LU name</td>
<td>3</td>
<td>VSELU</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>Remote Transaction Program name</td>
<td>4</td>
<td>MQ01</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>LU 6.2 CPI-C Side Information profile name</td>
<td></td>
<td>VSECPIC</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>Mode name</td>
<td>6</td>
<td>#INTER</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>LAN destination address</td>
<td>5</td>
<td>400074511092</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>Token-Ring Link Station profile name</td>
<td>6</td>
<td>VSEP0</td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>CP name of adjacent node</td>
<td>2</td>
<td>VSEP0</td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>LU 6.2 partner LU profile name</td>
<td></td>
<td>VSELUPRO</td>
<td></td>
</tr>
</tbody>
</table>

#### Explanation of terms

1. **Network name**
   - This is the unique ID of the network to which you are connected. Your network administrator will tell you this value.

2. **Control Point name**
   - This is a unique control point name for this workstation. Your network administrator will assign this to you.

3. **XID node ID**
   - This is a unique identifier for this workstation. On other platforms it is often referred to as the exchange ID (XID). Your network administrator will assign this to you.

4. **Local LU name**
   - A logical unit (LU) manages the exchange of data between systems. The local LU name is the name of the LU on your system. Your network administrator will assign this to you.
5 Local LU alias
The local LU alias is the name by which your local LU is known to your applications. You can choose this name yourself. It need be unique only on this machine.

6 TP Name
WebSphere MQ applications trying to converse with this workstation will specify a symbolic name for the program to be run at the receiving end. This will have been defined on the channel definition at the sender. It is recommended that when AIX is the receiver a Transaction Program Name of MQSERIES is used, or in the case of a connection to VSE/ESA, where the length is limited to 4 bytes, use MQTP.

See Table 20 on page 194 for more information.

7 Full path to TP executable
This is the path and name of a shell script file that invokes the actual program to be run when a conversation is initiated with this workstation. You can choose the path and name of the script file. The contents of the file are illustrated in “WebSphere MQ for AIX TPN setup” on page 213.

8 Token-ring adapter address
This is the 12-character hex address of the token-ring card. It can be found by entering the AIX command:

```
lsfg -v -l tokn
```

where n is the number assigned to the token-ring adapter you are using. The Network Address field of the Token-Ring section indicates the adapter’s address.

9 Mode name
This is the name of a configuration profile used by Communications Server for AIX. The profile contains the set of parameters that control the APPC conversation. The mode name specified in the profile will be assigned to you by your network administrator. You supply the name to be used for the profile.

13 LU 6.2 CPI-C Side Information profile name
This is a name given to the Side Information profile defining a partner node. You supply the name. It needs to be unique only on this machine. You will later use the name in the WebSphere MQ sender channel definition.

16 Token-Ring Link Station profile name
This is the name of a configuration profile used by Communications Server for AIX. You supply the name to be used for the profile. The link station profile associates the link station with the SNA DLC profile, which has been used to define the hardware adapter and link characteristics, and the node control point.

17 CP name of adjacent node
This is the unique control point name of the partner system with which you are establishing communication. Your network administrator will assign this to you.

18 LU 6.2 partner LU profile name
This is the name of a configuration profile used by Communications Server for AIX. You supply the name to be used for the profile. It needs to be unique only on this machine. The profile defines parameters for establishing a session with a specific partner LU. In some scenarios, this
AIX and LU 6.2

profile may not be required but it is shown here to reduce the likelihood of error. See the SNA Server for AIX Configuration Reference manual for details.
Establishing a session using Communications Server for AIX V5

Verify the level of Communications Server software you have installed by entering the AIX command:

```
lslpp -h sna.rte
```

The level displayed in the response needs to be at least Version 5.0.

To update the SNA configuration profile, you need root authority. (Without root authority you can display options and appear to modify them, but cannot actually make any changes.) You can make configuration changes when SNA is either active or inactive.

The configuration scenario that follows was accomplished using the graphical interface.

Note: The setup used is APPN using independent LUs.

If you are an experienced user of AIX, you may choose to circumvent the panels and use the command-line interface. Refer to the SNA Server for AIX Configuration Reference manual to see the commands that correspond to the panels illustrated.

Throughout the following example, only the panels for profiles that must be added or updated are shown.

Configuring your node

This configuration uses a token ring setup. To define the end node to connect to the network node (assuming that a network node already exists), you need to:

1. Click on Services from the main menu on the main window.
2. Select Configuration node parameters ... from the drop-down list. A window entitled Node parameters appears:

3. Click on End node for APPN support.
4. In the SNA addressing box, enter a name and alias for the Control point. The Control point name consists of a Network name (1) and a Control point name (2).
5. Enter the Node ID (3) of your local machine.
6. Click on OK.

You have now configured your node to connect to the network node.
Using Communications Server for AIX

Configuring connectivity to the network

1. Defining your port:
   a. From the main menu of the main window, click on Services, Connectivity, and New port ... A window entitled Add to machine name screen appears.
   b. Select the default card for connecting to the network (Token ring card).
   c. Click on OK. A window entitled Token ring SAP appears:

   ![Token ring SAP window]

   d. Enter a port name in the SNA port name box, for example, MQPORT.
   e. Check Initially Active.
   f. Click on OK.

2. Defining your connection to the network node:
   a. From the main menu on the main window, click on Services, Connectivity, and New link station ...
   b. Click on OK to link your station to the chosen port (MQPORT). A window entitled Token ring link station appears:
c. Enter a name for your link station ( ), for example, NETNODE.
d. Enter the port name to which you want to connect the link station. In this case, the port name would be MQPORT.
e. Check Any in the LU traffic box.
f. Define where the remote node is by entering the control point on the network node in the Independent LU traffic box. The control point consists of a Network name ( ) and a CP name of adjacent node ( ).

Note: The network node does not have to be on the remote system that you are connecting to.
g. Ensure the Remote node type is Network node.
h. In the Contact information, enter the MAC address ( ) of the token ring card on the network node.

Note: The network node does not have to be on the remote system that you are connecting to.
i. Click on Advanced .... A window entitled Token ring parameters appears:
j. Check Remote node is network node server.

k. Click on OK. The Token ring link station window remains on the screen.

l. Click on OK on the Token ring link station window.

**Defining a local LU**

To define a local LU:

1. From the main menu on the main window, click on Services, APPC, and New independent local LU ... A window entitled Local LU appears:

![Local LU window](image)

*Figure 33. Local LU window*

2. Enter an LU name (A) and alias (B).

3. Click on OK.

You have now set up a basic SNA system.

To define the mode controlling the SNA session limits:

1. From the main menu in the main window, click on Services, APPC, and Modes ...

2. Select the New ... button. A window entitled Mode appears:
3. Enter a Name (9) for your mode.
4. When you are happy with the session limits, click on OK. The Modes window remains on the screen.
5. Click on Done in the Modes window.

Defining a transaction program
WebSphere MQ allows you to use the Communications Server for AIX V5 graphical interface to configure transaction programs.

If you are migrating from a previous version of MQSeries, you should delete any existing Communications Server definitions of transaction programs that can be invoked by WebSphere MQ using the following commands:
1. Type
   
   snaadmin delete_tp_load_info.tp_name=xxxxx

2. Then type
   
   snaadmin delete_tp.tp_name=xxxxx

An attempt to invoke a previously defined transaction program results in a SNA sense code of 084B6031. In addition, error message AMQ9213 is returned. See [WebSphere MQ Messages](#) for more information about this and other WebSphere MQ messages.

You can then re-create the transaction program definition using the following instructions

From the main window, click Services, APPC, and Transaction programs ... The following panel is displayed:
1. Type **TP name** (6) in the **Application TP** field.
2. Clear the **Queue incoming Allocates** check box.
3. Type the **Full path to executable** (7).
4. Type **−m Local queue manager** in the **Arguments** field.
5. Type **mqm** in the **User ID** and **Group ID** fields.
6. Enter environment variables **APPCLLU=local LU (4)** and **APPCTPN=Invokable TP (6)** separated by the pipe character in the **Environment** field.
7. Click **OK**.

**Establishing a TCP connection**

1. Edit the file `/etc/services`.

   **Note:** To edit the `/etc/services` file, you must be logged in as a superuser or root. If you do not have the following line in that file, add it as shown:

   ```
   MQSeries 1414/tcp # MQSeries channel listener
   ```

2. Edit the file `/etc/inetd.conf`. If you do not have the following line in that file, add it as shown:

   ```
   MQSeries stream tcp nowait root /usr/mqm/bin/amqcrsta amqcrsta
   [-m queue.manager.name]
   ```

3. Enter the command `refresh -s inetd`.

   **Note:** You must add `root` to the mqm group. You need not have the primary group set to mqm. As long as mqm is in the set of groups, you can use the commands. If you are running only applications that use the queue manager you do not need mqm group authority.
What next?
The connection is now established. You are ready to complete the configuration. Go to "WebSphere MQ for AIX configuration".

Establishing a UDP connection

To establish a UDP connection, ensure a listener is started by issuing the following MQSC command:
runmqslr -m QMgrName -t UDP -p 1414

Notes:
1. You cannot start a listener channel on AIX using the START LISTENER MQSC command.
2. Using the runmqslr command implies that you must not add entries to the /etc/services and /etc/inetd.conf file for UDP on WebSphere MQ for AIX.

What next?
The connection is now established. You are ready to complete the configuration. Go to "WebSphere MQ for AIX configuration".

WebSphere MQ for AIX configuration

Notes:
1. Before beginning the installation process ensure that you have first created the mqm user and group, and set the password.
2. If installation fails as a result of insufficient space in the file system you can increase the size as follows, using the command smit C sna. (Use df to display the current status of the file system. This will indicate the logical volume that is full.)
   -- Physical and Logical Storage
   -- File Systems
   -- Add / Change / Show / Delete File Systems
   -- Journaled File Systems
   -- Change/Show Characteristics of a Journaled File System
3. Start any channel using the command:
   runmqchl -c channel.name
4. Sample programs are installed in /usr/mqm/samp.
5. Error logs are stored in /var/mqm/qmgrs/qmgrname/errors.
6. You can start an AIX trace of the WebSphere MQ components using the command:
   trace -a -j30D,30E -o trace.file
   You can stop AIX trace by entering:
   trcstop
   Format the trace report using the command:
   trcrpt -t /usr/mqm/lib/amqtrc.fmt trace.file > trace.report
7. When you are using the command interpreter runmqsc to enter administration commands, a + at the end of a line indicates that the next line is a continuation. Ensure that there is a space between the last parameter and the continuation character.
AIX configuration

Basic configuration

1. Create the queue manager from the AIX command line using the command:
   
   ```bash
crtmqm -u dlqname -q aix
   ```

   where:
   
   - `aix`  Is the name of the queue manager
   - `-q` Indicates that this is to become the default queue manager
   - `-u dlqname` Specifies the name of the undeliverable message queue

   This command creates a queue manager and a set of default objects.

2. Start the queue manager from the AIX command line using the command:
   
   ```bash
   strmqm aix
   ```

   where `aix` is the name given to the queue manager when it was created.

3. Start `runmqsc` from the AIX command line and use it to create the undeliverable message queue by entering the command:
   
   ```bash
   def q1 (dlqname)
   ```

   where `dlqname` is the name given to the undeliverable message queue when the queue manager was created.

Channel configuration

The following section details the configuration to be performed on the AIX queue manager to implement the channel described in Figure 32 on page 101.

In each case the MQSC command is shown. Either start `runmqsc` from an AIX command line and enter each command in turn, or build the commands into a command file.

Examples are given for connecting WebSphere MQ for AIX and MQSeries for OS/2 Warp. If you wish to connect to another WebSphere MQ product use the appropriate set of values from the table in place of those for OS/2.

Note: The words in **bold** are user-specified and reflect the names of WebSphere MQ objects used throughout these examples. If you change the names used here, ensure that you also change the other references made to these objects throughout this book. All others are keywords and should be entered as shown.

Table 22. Configuration worksheet for WebSphere MQ for AIX

<table>
<thead>
<tr>
<th>ID</th>
<th>Parameter Name</th>
<th>Reference</th>
<th>Example Used</th>
<th>User Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>Queue Manager Name</td>
<td>AIX</td>
<td></td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>Local queue name</td>
<td>AIX.LOCALQ</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Connection to MQSeries for OS/2 Warp

The values in this section of the table must match those used in Table 16 on page 161 as indicated.

| C  | Remote queue manager name       | A             | OS2                |                |
| D  | Remote queue name               | A             | OS2.REMOTEQ        |                |
| E  | Queue name at remote system     | A             | OS2.LOCALQ         |                |
### Table 22. Configuration worksheet for WebSphere MQ for AIX (continued)

<table>
<thead>
<tr>
<th>ID</th>
<th>Parameter Name</th>
<th>Reference</th>
<th>Example Used</th>
<th>User Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>F</td>
<td>Transmission queue name</td>
<td>OS2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>G</td>
<td>Sender (SNA) channel name</td>
<td>AIX.OS2.SNA</td>
<td></td>
<td></td>
</tr>
<tr>
<td>H</td>
<td>Sender (TCP/IP) channel name</td>
<td>AIX.OS2.TCP</td>
<td></td>
<td></td>
</tr>
<tr>
<td>I</td>
<td>Receiver (SNA) channel name</td>
<td>OS2.AIX.SNA</td>
<td></td>
<td></td>
</tr>
<tr>
<td>J</td>
<td>Receiver (TCP/IP) channel name</td>
<td>OS2.AIX.TCP</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Connection to WebSphere MQ for Windows**

The values in this section of the table must match those used in Table 18 on page 184 as indicated.

| C  | Remote queue manager name                   | WINNT              |                |                             |
| D  | Remote queue name                           | WINNT.REMOTEQ      |                |                             |
| E  | Queue name at remote system                 | WINNT.LOCALQ       |                |                             |
| F  | Transmission queue name                     | WINNT              |                |                             |
| G  | Sender (SNA) channel name                   | AIX.WINNT.SNA      |                |                             |
| H  | Sender (TCP/IP) channel name                | AIX.WINNT.TCP      |                |                             |
| I  | Receiver (SNA) channel name                 | WINNT.AIX.SNA      |                |                             |
| J  | Receiver (TCP) channel name                 | WINNT.AIX.TCP      |                |                             |

**Connection to MQSeries for Compaq Tru64 UNIX**

The values in this section of the table must match those used in Table 23 on page 216 as indicated.

| C  | Remote queue manager name                   | DECUX              |                |                             |
| D  | Remote queue name                           | DECUX.REMOTEQ      |                |                             |
| E  | Queue name at remote system                 | DECUX.LOCALQ       |                |                             |
| F  | Transmission queue name                     | DECUX              |                |                             |
| H  | Sender (TCP) channel name                   | DECUX.AIX.TCP      |                |                             |
| J  | Receiver (TCP) channel name                 | AIX.DECUX.TCP      |                |                             |

**Connection to WebSphere MQ for HP-UX**

The values in this section of the table must match those used in Table 25 on page 239 as indicated.

| C  | Remote queue manager name                   | HPUX               |                |                             |
| D  | Remote queue name                           | HPUX.REMOTEQ       |                |                             |
| E  | Queue name at remote system                 | HPUX.LOCALQ        |                |                             |
| F  | Transmission queue name                     | HPUX               |                |                             |
| G  | Sender (SNA) channel name                   | AIX.HPUX.SNA       |                |                             |
| H  | Sender (TCP) channel name                   | AIX.HPUX.TCP       |                |                             |
| I  | Receiver (SNA) channel name                 | HPUX.AIX.SNA       |                |                             |
| J  | Receiver (TCP) channel name                 | HPUX.AIX.TCP       |                |                             |

**Connection to MQSeries for AT&T GIS UNIX**

The values in this section of the table must match those used in Table 27 on page 253 as indicated.

<p>| C  | Remote queue manager name                   | GIS                |                |                             |
| D  | Remote queue name                           | GIS.REMOTEQ        |                |                             |
| E  | Queue name at remote system                 | GIS.LOCALQ         |                |                             |
| F  | Transmission queue name                     | GIS                |                |                             |
| G  | Sender (SNA) channel name                   | AIX.GIS.SNA        |                |                             |
| H  | Sender (TCP) channel name                   | AIX.GIS.TCP        |                |                             |
| I  | Receiver (SNA) channel name                 | GIS.AIX.SNA        |                |                             |
| J  | Receiver (TCP/IP) channel name              | GIS.AIX.TCP        |                |                             |</p>
<table>
<thead>
<tr>
<th>ID</th>
<th>Parameter Name</th>
<th>Reference</th>
<th>Example Used</th>
<th>User Value</th>
</tr>
</thead>
</table>

### Connection to WebSphere MQ for Solaris

The values in this section of the table must match those used in Table 30 on page 290 as indicated.

<table>
<thead>
<tr>
<th></th>
<th>Remote queue manager name</th>
<th>SOLARIS</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>D</td>
<td>Remote queue name</td>
<td>SOLARIS.REMOTEQ</td>
<td></td>
<td></td>
</tr>
<tr>
<td>E</td>
<td>Queue name at remote system</td>
<td>SOLARIS.LOCALQ</td>
<td></td>
<td></td>
</tr>
<tr>
<td>F</td>
<td>Transmission queue name</td>
<td>SOLARIS</td>
<td></td>
<td></td>
</tr>
<tr>
<td>G</td>
<td>Sender (SNA) channel name</td>
<td>AIX.SOLARIS.SNA</td>
<td></td>
<td></td>
</tr>
<tr>
<td>H</td>
<td>Sender (TCP/IP) channel name</td>
<td>AIX.SOLARIS.TCP</td>
<td></td>
<td></td>
</tr>
<tr>
<td>I</td>
<td>Receiver (SNA) channel name</td>
<td>SOLARIS.AIX.SNA</td>
<td></td>
<td></td>
</tr>
<tr>
<td>J</td>
<td>Receiver (TCP/IP) channel name</td>
<td>SOLARIS.AIX.TCP</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Connection to WebSphere MQ for iSeries

The values in this section of the table must match those used in Table 49 on page 535 as indicated.

|   | Remote queue manager name      | AS400                            |                   |                   |
| D | Remote queue name              | AS400.REMOTEQ                    |                   |                   |
| E | Queue name at remote system    | AS400.LOCALQ                     |                   |                   |
| F | Transmission queue name        | AS400                            |                   |                   |
| G | Sender (SNA) channel name      | AIX.AS400.SNA                    |                   |                   |
| H | Sender (TCP) channel name      | AIX.AS400.TCP                    |                   |                   |
| I | Receiver (SNA) channel name    | AS400.AIX.SNA                    |                   |                   |
| J | Receiver (TCP) channel name    | AS400.AIX.TCP                    |                   |                   |

### Connection to WebSphere MQ for z/OS without CICS

The values in this section of the table must match those used in Table 34 on page 372 as indicated.

|   | Remote queue manager name      | MVS                             |                   |                   |
| D | Remote queue name              | MVS.REMOTEQ                     |                   |                   |
| E | Queue name at remote system    | MVS.LOCALQ                      |                   |                   |
| F | Transmission queue name        | MVS                             |                   |                   |
| G | Sender (SNA) channel name      | AIX.MVS.SNA                     |                   |                   |
| H | Sender (TCP) channel name      | AIX.MVS.TCP                     |                   |                   |
| I | Receiver (SNA) channel name    | MVS.AIX.SNA                     |                   |                   |
| J | Receiver (TCP) channel name    | MVS.AIX.TCP                     |                   |                   |

### Connection to WebSphere MQ for z/OS using queue-sharing groups

The values in this section of the table must match those used in Table 43 on page 454 as indicated.

|   | Remote queue manager name      | QSG                             |                   |                   |
| D | Remote queue name              | QSG.REMOTEQ                     |                   |                   |
| E | Queue name at remote system    | QSG.SHAREDQ                     |                   |                   |
| F | Transmission queue name        | QSG                             |                   |                   |
| G | Sender (SNA) channel name      | AIX.QSG.SNA                     |                   |                   |
| H | Sender (TCP) channel name      | AIX.QSG.TCP                     |                   |                   |
| I | Receiver (SNA) channel name    | QSG.AIX.SNA                     |                   |                   |
| J | Receiver (TCP) channel name    | QSG.AIX.TCP                     |                   |                   |

### Connection to MQSeries for VSE/ESA

The values in this section of the table must match those used in Table 51 on page 554 as indicated.

|   | Remote queue manager name      | VSE                             |                   |                   |
### WebSphere MQ for AIX sender-channel definitions using SNA

```bash
def ql (OS2) +
  usage(xmitq) +
replace

def qr (OS2.REMOTEQ) +
  rname(OS2.LOCALQ) +
  rmqname(OS2) +
  xmitq(OS2) +
replace

def chl (AIX.OS2.SNA) chltype(sdr) +
  trptype(lu62) +
  connname('OS2CPIC') +
  xmitq(OS2) +
replace
```

### WebSphere MQ for AIX receiver-channel definitions using SNA

```bash
def ql (AIX.LOCALQ) replace

def chl (OS2.AIX.SNA) chltype(rcvr) +
  trptype(lu62) +
replace
```

### WebSphere MQ for AIX TPN setup

During the AIX Communications Server configuration process, an LU 6.2 TPN profile was created, which contained the full path to a TP executable. In the example the file was called u/interops/AIX.crs6a. You can choose a name, but you are recommended to include the name of your queue manager in it. The contents of the executable file must be:

```bash
#!/bin/sh
/opt/mqm/bin/amqcrs6a -m aix
```

where `aix` is the queue manager name (A). After creating this file, enable it for execution by running the command:

```bash
chmod 755 /u/interops/AIX.crs6a
```

As an alternative to creating an executable file, you can specify the path on the Add LU 6.2 TPN Profile panel, using command line parameters.

Specifying a path in one of these two ways ensures that SNA receiver channels activate correctly when a sender channel initiates a conversation.

### WebSphere MQ for AIX sender-channel definitions using TCP

```bash
def ql (OS2) +
  usage(xmitq) +
replace

def qr (OS2.REMOTEQ) +
  rname(OS2.LOCALQ) +
```

---

**Table 22. Configuration worksheet for WebSphere MQ for AIX (continued)**

<table>
<thead>
<tr>
<th>ID</th>
<th>Parameter Name</th>
<th>Reference</th>
<th>Example Used</th>
<th>User Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>D</td>
<td>Remote queue name</td>
<td></td>
<td>VSE.REMOTEQ</td>
<td></td>
</tr>
<tr>
<td>E</td>
<td>Queue name at remote system</td>
<td>E</td>
<td>VSE.LOCALQ</td>
<td></td>
</tr>
<tr>
<td>F</td>
<td>Transmission queue name</td>
<td></td>
<td>VSE</td>
<td></td>
</tr>
<tr>
<td>G</td>
<td>Sender channel name</td>
<td></td>
<td>AIX.VSE.SNA</td>
<td></td>
</tr>
<tr>
<td>I</td>
<td>Receiver channel name</td>
<td></td>
<td>VSE.AIX.SNA</td>
<td></td>
</tr>
</tbody>
</table>
AIX configuration

```
rmqname(OS2) +
xmitq(OS2) +
replace

def chl (AIX.OS2.TCP) chltype(sdr) +
   trptype(tcp) +
   conname(remote_tcpip_hostname) +
   xmitq(OS2) +
   replace

WebSphere MQ for AIX receiver-channel definitions using TCP

def q1 (AIX.LOCALQ) replace

def chl (OS2.AIX.TCP) chltype(rcvr) +
   trptype(tcp) +
   replace

WebSphere MQ for AIX sender-channel definitions using UDP

def q1 (OS2) +
   usage(xmitq) +
   replace

def qr (OS2.REMOTEQ) +
   rname(OS2.LOCALQ) +
   rmqname(OS2) +
   xmitq(OS2) +
   replace

def chl (AIX.OS2.UDP) chltype(sdr) +
   trptype(udp) +
   conname(remote_udpip_hostname) +
   xmitq(OS2) +
   replace

WebSphere MQ for AIX receiver-channel definitions using UDP

def q1 (AIX.LOCALQ) replace

def chl (OS2.AIX.UDP) chltype(rcvr) +
   trptype(udp) +
   replace
```
Chapter 15. Example configuration - IBM MQSeries for Compaq Tru64 UNIX

This chapter shows how to set up TCP communication links from MQSeries for Compaq Tru64 UNIX to WebSphere MQ products on other platforms.

**Note:** MQSeries for Compaq Tru64 UNIX supports the TCP communication protocol only.

Once the connection is established, you need to define some channels to complete the configuration. This process is described in "MQSeries for Compaq Tru64 UNIX configuration".

See Chapter 7, “Example configuration chapters in this book” on page 101 for background information about this chapter and how to use it.

Establishing a TCP connection

1. Edit the file /etc/services.
   - **Note:** To edit the /etc/services file, you must be logged in as a superuser or root. If you do not have the following line in that file, add it as shown:
     ```
     MQSeries 1414/tcp # MQSeries channel listener
     ```

2. Edit the file /etc/inetd.conf. If you do not have the following line in that file, add it as shown:
   ```
   MQSeries stream tcp nowait root /opt/mqm/bin/amqcrsta amqcrsta [-m queue.manager.name]
   ```

3. Find the process ID of the inetd with the command:
   ```
   ps -ef | grep inetd
   ```

4. Run the command:
   ```
   kill -1 inetd processid
   ```

What next?

Inetd is now ready to listen for incoming requests and pass them to WebSphere MQ. You are ready to complete the configuration as described in the next section.

MQSeries for Compaq Tru64 UNIX configuration

Before beginning the installation process ensure that you have first created the mqm user and group, and set the password.

Start any channel using the command:
```
runmqch1 -c channel.name -m qmname
```

**Notes:**
1. Sample programs are installed in /opt/mqm/samp.
2. Error logs are stored in /var/mqm/qmgrs/qmname/errors.

© Copyright IBM Corp. 1993, 2002
Compaq Tru64 UNIX configuration

3. When you are using the command interpreter runmqsc to enter administration commands, a + at the end of a line indicates that the next line is a continuation. Ensure that there is a space between the last parameter and the continuation character.

Basic configuration

1. Create the queue manager from the UNIX prompt using the command:
   
   ```
crtmqm -u d'lnqueanme -q qmname
   ```
   
   where:
   
   - `qmname` is the name of the queue manager
   - `-u` indicates that this is to become the default queue manager
   - `-q d'lnqueanme` specifies the name of the undeliverable message queue

   This command creates a queue manager and sets the DEADQ attribute of the queue manager, but does not create the undeliverable message queue.

2. Start the queue manager from the UNIX prompt using the command:
   
   ```
strmqm qmname
   ```
   
   where `qmname` is the name given to the queue manager when it was created.

Channel configuration

This section describes the configuration to be performed on the Compaq Tru64 UNIX queue manager to implement the single channel, and the WebSphere MQ objects associated with it.

Examples are given at the end of this chapter for connecting MQSeries for Compaq Tru64 UNIX and MQSeries for OS/2 Warp. If you wish to connect to another WebSphere MQ product use the appropriate set of values from the table in place of those for OS/2.

In each example, the MQSC command is shown. Either start runmqsc from a UNIX prompt and enter each command in turn, or build the commands into a command file.

Note: The words in **bold** are user-specified and reflect the names of WebSphere MQ objects used throughout these examples. All others are keywords and should be entered as shown.

<p>| Table 23. Configuration worksheet for MQSeries for Compaq Tru64 UNIX |
|--------------------------|-----------------------------|----------------------|------------------|</p>
<table>
<thead>
<tr>
<th>ID</th>
<th>Parameter Name</th>
<th>Reference</th>
<th>Example Used</th>
<th>User Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>Queue Manager Name</td>
<td></td>
<td>DECUX</td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>Local queue name</td>
<td></td>
<td>DECUX.LOCALQ</td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>Remote queue manager name</td>
<td>A</td>
<td>OS2</td>
<td></td>
</tr>
<tr>
<td>D</td>
<td>Remote queue name</td>
<td></td>
<td>OS2.REMOTEQ</td>
<td></td>
</tr>
<tr>
<td>E</td>
<td>Queue name at remote system</td>
<td>B</td>
<td>OS2.LOCALQ</td>
<td></td>
</tr>
<tr>
<td>F</td>
<td>Transmission queue name</td>
<td></td>
<td>OS2</td>
<td></td>
</tr>
</tbody>
</table>
Table 23. Configuration worksheet for MQSeries for Compaq Tru64 UNIX (continued)

<table>
<thead>
<tr>
<th>ID</th>
<th>Parameter Name</th>
<th>Reference</th>
<th>Example Used</th>
<th>User Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>Sender (TCP) channel name</td>
<td>DECUX.OS2.TCP</td>
<td></td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>Receiver (TCP) channel name</td>
<td>OS2.DECUX.TCP</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Connection to WebSphere MQ for Windows

The values in this section of the table must match those used in Table 18 on page 184.

| C  | Remote queue manager name | WINNT         |              |            |
| D  | Remote queue name         | WINNT.REMOTEQ |              |            |
| E  | Queue name at remote system | WINNT.LOCALQ |              |            |
| F  | Transmission queue name   | WINNT         |              |            |
| H  | Sender (TCP) channel name | DECUX.WINNT.TCP |          |            |
| J  | Receiver (TCP) channel name | WINNT.DECUX.TCP |        |            |

Connection to WebSphere MQ for AIX

The values in this section of the table must match those used in Table 22 on page 210.

| C  | Remote queue manager name | AIX           |              |            |
| D  | Remote queue name         | AIX.REMOTEQ   |              |            |
| E  | Queue name at remote system | AIX.LOCALQ |              |            |
| F  | Transmission queue name   | AIX           |              |            |
| H  | Sender (TCP) channel name | DECUX.AIX.TCP |              |            |
| J  | Receiver (TCP) channel name | AIX.DECUX.TCP |          |            |

Connection to MQSeries for AT&T GIS UNIX

The values in this section of the table must match those used in Table 27 on page 253.

| C  | Remote queue manager name | GIS           |              |            |
| D  | Remote queue name         | GIS.REMOTEQ   |              |            |
| E  | Queue name at remote system | GIS.LOCALQ |              |            |
| F  | Transmission queue name   | GIS           |              |            |
| H  | Sender (TCP) channel name | DECUX.GIS.TCP |              |            |
| J  | Receiver (TCP) channel name | GIS.DECUX.TCP |          |            |

Connection to WebSphere MQ for Solaris

The values in this section of the table must match those used in Table 30 on page 290.

| C  | Remote queue manager name | SOLARIS       |              |            |
| D  | Remote queue name         | SOLARIS.REMOTEQ |          |            |
| E  | Queue name at remote system | SOLARIS.LOCALQ |          |            |
| F  | Transmission queue name   | SOLARIS       |              |            |
| H  | Sender (TCP) channel name | DECUX.SOLARIS.TCP |        |            |
| J  | Receiver (TCP) channel name | SOLARIS.DECUX.TCP |        |            |

Connection to WebSphere MQ for iSeries.

The values in this section of the table must match those used in Table 49 on page 535.

| C  | Remote queue manager name | AS400         |              |            |
| D  | Remote queue name         | AS400.REMOTEQ |              |            |
| E  | Queue name at remote system | AS400.LOCALQ |              |            |
| F  | Transmission queue name   | AS400         |              |            |
| H  | Sender (TCP) channel name | DECUX.AS400.TCP |          |            |
| J  | Receiver (TCP) channel name | AS400.DECUX.TCP |          |            |
## MQSeries for Compaq Tru64 UNIX configuration

**Table 23. Configuration worksheet for MQSeries for Compaq Tru64 UNIX (continued)**

<table>
<thead>
<tr>
<th>ID</th>
<th>Parameter Name</th>
<th>Reference</th>
<th>Example Used</th>
<th>User Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>Remote queue manager name</td>
<td>MVS</td>
<td></td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>Remote queue name</td>
<td>MVS.REMOTEQ</td>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>Queue name at remote system</td>
<td>MVS.LOCALQ</td>
<td></td>
<td></td>
</tr>
<tr>
<td>D</td>
<td>Transmission queue name</td>
<td>MVS</td>
<td></td>
<td></td>
</tr>
<tr>
<td>E</td>
<td>Sender (TCP) channel name</td>
<td>DECUX.MVS.TCP</td>
<td></td>
<td></td>
</tr>
<tr>
<td>F</td>
<td>Receiver (TCP) channel name</td>
<td>MVS.DECUX.TCP</td>
<td></td>
<td></td>
</tr>
<tr>
<td>G</td>
<td>Remote queue manager name</td>
<td>VSE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>H</td>
<td>Remote queue name</td>
<td>VSE.REMOTEQ</td>
<td></td>
<td></td>
</tr>
<tr>
<td>I</td>
<td>Queue name at remote system</td>
<td>VSE.LOCALQ</td>
<td></td>
<td></td>
</tr>
<tr>
<td>J</td>
<td>Transmission queue name</td>
<td>VSE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>K</td>
<td>Sender (SNA) channel name</td>
<td>DECUX.VSE.SNA</td>
<td></td>
<td></td>
</tr>
<tr>
<td>L</td>
<td>Receiver (SNA) channel name</td>
<td>VSE.DECUX.SNA</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Connection to WebSphere MQ for z/OS without CICS**

The values in this section of the table must match those used in Table 34 on page 372.

**MQSeries for Compaq Tru64 UNIX sender-channel definitions using TCP/IP**

```
def ql (OS2) +
  usage(xmitq) +
  replace
```

```
def qr (OS2.REMOTEQ) +
  rname(OS2.LOCALQ) +
  rqmname(OS2) +
  xmitq(OS2) +
  replace
```

```
def chl (DECUX.OS2.TCP) chltype(sdr) +
  trptype(tcp) +
  conname(remote_tcpip_hostname) +
  xmitq(OS2) +
  replace
```

**MQSeries for Compaq Tru64 UNIX receiver-channel definitions using TCP/IP**

```
def qr (DECUX.LOCALQ) replace
```

```
def chl (OS2.DECUX.TCP) chltype(rcvr) +
  trptype(tcp) +
  replace
```
Chapter 16. Example configuration - IBM WebSphere MQ for HP-UX

This chapter gives an example of how to set up communication links from WebSphere MQ for HP-UX to WebSphere MQ products on the following platforms:

- OS/2
- Windows
- AIX
- Compq Tru 64UNIX
- AT&T GIS UNIX
- Solaris
- iSeries
- z/OS without CICS
- VSE/ESA

First it describes the parameters needed for an LU 6.2 connection, then it describes “Establishing a session using HP SNAplus2” on page 223 and “Establishing a TCP connection” on page 237.

Once the connection is established, you need to define some channels to complete the configuration. This is described in “WebSphere MQ for HP-UX configuration” on page 238.

See Chapter 7, “Example configuration chapters in this book” on page 101 for background information about this chapter and how to use it.

Configuration parameters for an LU 6.2 connection

Table 24 presents a worksheet listing all the parameters needed to set up communication from HP-UX to one of the other WebSphere MQ platforms. The worksheet shows examples of the parameters, which have been tested in a working environment, and leaves space for you to fill in your own values. An explanation of the parameter names follows the worksheet. Use the worksheet in this chapter in conjunction with the worksheet in the chapter for the platform to which you are connecting.

Configuration worksheet

Use this worksheet to record the values you use for your configuration. Where numbers appear in the Reference column they indicate that the value must match that in the appropriate worksheet elsewhere in this book. The examples that follow in this chapter refer back to the values in the ID column. The entries in the Parameter Name column are explained in “Explanation of terms” on page 222.

Table 24. Configuration worksheet for HP SNAplus2

<table>
<thead>
<tr>
<th>ID</th>
<th>Parameter Name</th>
<th>Reference</th>
<th>Example</th>
<th>User Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Configuration file name</td>
<td>sna_node.cfg</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Control point name</td>
<td></td>
<td>HPUXPU</td>
<td></td>
</tr>
</tbody>
</table>

4. This platform has become NCR UNIX SVR4 MP-RAS, R3.0
## HP-UX and LU 6.2

**Table 24. Configuration worksheet for HP SNAplus2 (continued)**

<table>
<thead>
<tr>
<th>ID</th>
<th>Parameter Name</th>
<th>Reference</th>
<th>Example</th>
<th>User Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>Node ID to send</td>
<td></td>
<td>05D 54321</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Network name</td>
<td></td>
<td>NETID</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Local APPC LU</td>
<td></td>
<td>HPUXLU</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>APPC mode</td>
<td></td>
<td>#INTER</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>Invokable TP</td>
<td></td>
<td>MQSERIES</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>Token-Ring adapter address</td>
<td></td>
<td>100090DC2C7C</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>Port name</td>
<td></td>
<td>MQPORT</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>Full path to executable</td>
<td></td>
<td>/opt/mqm/bin/amqrs6a</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>Local queue manager</td>
<td></td>
<td>hpux</td>
<td></td>
</tr>
</tbody>
</table>

**Connection to an OS/2 system**

The values in this section of the table must match those used in Table 15 on page 144, as indicated.

| 12 | Link station name             | OS2CONN  |          |              |
| 13 | Network name                  | NETID    |          |              |
| 14 | CP name                       | OS2PU    |          |              |
| 15 | Remote LU                     | OS2LU    |          |              |
| 16 | Application TP                | MQSERIES |          |              |
| 17 | Mode name                     | #INTER   |          |              |
| 18 | CPI-C symbolic destination name| OS2CPIC |          |              |
| 19 | Remote network address        | 10005AFC5D83 |          |              |
| 20 | Node ID to receive            | 05D 12345 |          |              |

**Connection to a Windows system**

The values in this section of the table must match those used in Table 17 on page 168, as indicated.

| 12 | Link station name             | NTCONN   |          |              |
| 13 | Network name                  | NETID    |          |              |
| 14 | CP name                       | WINNTCP  |          |              |
| 15 | Remote LU                     | WINNTLU  |          |              |
| 16 | Application TP                | MQSERIES |          |              |
| 17 | Mode name                     | #INTER   |          |              |
| 18 | CPI-C symbolic destination name| NTCPIC |          |              |
| 19 | Remote network address        | 08005A5FAB9 |          |              |
| 20 | Node ID to receive            | 05D 12345 |          |              |

**Connection to an AIX system**

The values in this section of the table must match those used in Table 21 on page 197, as indicated.

| 12 | Link station name             | AIXCONN  |          |              |
| 13 | Network name                  | NETID    |          |              |
| 14 | CP name                       | AIXPU    |          |              |
| 15 | Remote LU                     | AIXLU    |          |              |
| 16 | Application TP                | MQSERIES |          |              |
| 17 | Mode name                     | #INTER   |          |              |
| 18 | CPI-C symbolic destination name| AIXCPIC |          |              |
| 19 | Remote network address        | 123456789012 |          |              |
| 20 | Node ID to receive            | 071 23456 |          |              |

**Connection to an AT&T GIS UNIX system**

The values in this section of the table must match those used in the table Table 26 on page 243, as indicated.
Table 24. Configuration worksheet for HP SNAplus2 (continued)

<table>
<thead>
<tr>
<th>ID</th>
<th>Parameter Name</th>
<th>Reference</th>
<th>Example</th>
<th>User Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>12</td>
<td>Link station name</td>
<td>GISCONN</td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>Network name</td>
<td>NETID</td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>CP name</td>
<td>GISPU</td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>Remote LU</td>
<td>GISLU</td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>Application TP</td>
<td>MQSERIES</td>
<td></td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>Mode name</td>
<td>#INTER</td>
<td></td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>CPI-C symbolic destination name</td>
<td>GISCPLIC</td>
<td></td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>Remote network address</td>
<td>10007038E86B</td>
<td></td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>Node ID to receive</td>
<td>03E 00018</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Connection to a Solaris system

The values in this section of the table must match those used in Table 28 on page 258, as indicated.

<table>
<thead>
<tr>
<th>ID</th>
<th>Parameter Name</th>
<th>Reference</th>
<th>Example</th>
<th>User Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>12</td>
<td>Link station name</td>
<td>SOLCONN</td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>Network name</td>
<td>NETID</td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>CP name</td>
<td>SOLARPU</td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>Remote LU</td>
<td>SOLARLU</td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>Application TP</td>
<td>MQSERIES</td>
<td></td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>Mode name</td>
<td>#INTER</td>
<td></td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>CPI-C symbolic destination name</td>
<td>SOLCPIC</td>
<td></td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>Remote network address</td>
<td>08002071CC8A</td>
<td></td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>node ID to receive</td>
<td>05D 310D6</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Connection to an iSeries system

The values in this section of the table must match those used in Table 48 on page 521, as indicated.

<table>
<thead>
<tr>
<th>ID</th>
<th>Parameter Name</th>
<th>Reference</th>
<th>Example</th>
<th>User Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>12</td>
<td>Link station name</td>
<td>AS4CONN</td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>Network name</td>
<td>NETID</td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>CP name</td>
<td>AS400PU</td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>Remote LU</td>
<td>AS400LU</td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>Application TP</td>
<td>MQSERIES</td>
<td></td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>Mode name</td>
<td>#INTER</td>
<td></td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>CPI-C symbolic destination name</td>
<td>AS4CPIC</td>
<td></td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>Remote network address</td>
<td>10005A5962EF</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Connection to a z/OS system without CICS

The values in this section of the table must match those used in Table 33 on page 366, as indicated.

<table>
<thead>
<tr>
<th>ID</th>
<th>Parameter Name</th>
<th>Reference</th>
<th>Example</th>
<th>User Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>12</td>
<td>Link station name</td>
<td>MVSCONN</td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>Network name</td>
<td>NETID</td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>CP name</td>
<td>MVSPU</td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>Remote LU</td>
<td>MVSLU</td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>Application TP</td>
<td>MQSERIES</td>
<td></td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>Mode name</td>
<td>#INTER</td>
<td></td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>CPI-C symbolic destination name</td>
<td>MVSCPIC</td>
<td></td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>Remote network address</td>
<td>400074511092</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Connection to a VSE/ESA system

The values in this section of the table must match those used in Table 50 on page 549, as indicated.

<table>
<thead>
<tr>
<th>ID</th>
<th>Parameter Name</th>
<th>Reference</th>
<th>Example</th>
<th>User Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>12</td>
<td>Link station name</td>
<td>VSECONN</td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>Network name</td>
<td>NETID</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Table 24. Configuration worksheet for HP SNAplus2 (continued)

<table>
<thead>
<tr>
<th>ID</th>
<th>Parameter Name</th>
<th>Reference</th>
<th>Example</th>
<th>User Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>14</td>
<td>CP name</td>
<td>2</td>
<td>VSEPU</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>Remote LU</td>
<td>3</td>
<td>VSELU</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>Application TP</td>
<td>4</td>
<td>MQ01</td>
<td>MQ01</td>
</tr>
<tr>
<td>17</td>
<td>Mode name</td>
<td></td>
<td>#INTER</td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>CPI-C symbolic destination name</td>
<td></td>
<td>VSECPC</td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>Remote network address</td>
<td>5</td>
<td>400074511092</td>
<td></td>
</tr>
</tbody>
</table>

**Explanation of terms**

1. **Configuration file name**
   This is the unique name of the SNAplus2 configuration file. The default for this name is `sna_node.cfg`.

   Although it is possible to edit this file it is strongly recommended that configuration is done using xsnapadmin.

2. **Control point name**
   This is the unique Control point name for this workstation. In the SNA network, the Control point is an addressable location (PU type 2.1). Your network administrator will assign this to you.

3. **Node ID to send**
   This is the unique ID of this workstation. On other platforms this is often referred to as the Exchange ID or XID. Your network administrator will assign this ID for you.

4. **Network name**
   This is the unique ID of the network to which you are connected. It is an alphanumeric value and can be 1-8 characters long. The network name works with the Control point name to uniquely identify a system. Your network administrator will tell you the value.

5. **Local APPC LU**
   An LU manages the exchange of data between transactions. The local APPC LU name is the name of the LU on your system. Your network administrator will assign this to you.

6. **APPC mode**
   This is the name given to the set of parameters that control the APPC conversation. This name must be defined at each partner system. Your network administrator will assign this to you.

7. **Invokable TP**
   WebSphere MQ applications trying to converse with this workstation will specify a symbolic name for the program to be run at the receiving end. This will have been defined on the channel definition at the sender. For simplicity, wherever possible use a transaction program name of MQSERIES, or in the case of a connection to VSE/ESA, where the length is limited to 4 bytes, use MQTP.

   See Table 20 on page 194 for more information.

8. **Token-ring adapter address**
   Use the HP-UX System Administration Manager (SAM) to discover the adapter address for this workstation. You need root authority to use SAM. From the initial menu, select **Networking and Communications**, then
select **Network Interface cards** followed by **LAN 0** (or whichever LAN you are using). The adapter address is displayed under the heading **Station Address** (hex). The card name represents the appropriate card type. If you do not have root level authority, your HP-UX system administrator can tell you the value.

9 **Port name**
This is a meaningful symbolic name that is used to associate the definitions with a network interface (in this case, a Token-Ring adapter). A separate Port must be defined for each physical device attached to the workstation.

10 **Full path to executable**
On HP SNAplus2 Release 5, this is the path and name of a shell script file that invokes the actual program to be run when a conversation is initiated with this workstation. You can choose the path and name of the script file. The contents of the file are illustrated in “WebSphere MQ for HP-UX invokable TP setup” on page 241. On HP SNAplus2 Release 6, this is the path and name of the program to be run when a conversation is initiated with this workstation. You enter the path in the TP invocation screen (see “Adding a TP definition using HP SNAplus2 Release 6” on page 235).

11 **Local queue manager**
This is the name of the queue manager on your local system.

10 **Link station name**
This is a meaningful symbolic name by which the connection to a peer or host node is known. It defines a logical path to the remote system. Its name is used only inside SNAplus2 and is specified by you. The connection must be associated with an existing Link and owned by one local node. You must define one connection for each partner or host system.

18 **CPI-C symbolic destination name**
This is a name given to the definition of a partner node. You choose the name. It need be unique only on this machine. Later you can use this name in the WebSphere MQ sender channel definition.

20 **Node ID to receive**
This is the unique ID of the partner workstation with which you will be communicating. On other platforms this is often referred to as the **Exchange ID** or **XID**. For a connection to a host system any values except 000 FFFFF and FFF FFFFF may be specified. Your network administrator will assign this ID for you.

### Establishing a session using HP SNAplus2

The following information guides you through the tasks you must perform to create the SNA infrastructure that WebSphere MQ requires. This example creates the definitions for a partner node and LU on OS/2.

Use **snap start** followed by **xsnapadmin** to enter the HP SNAplus2 configuration panels. You need root authority to use **xsnapadmin**.

### SNAplus2 configuration

SNAplus2 configuration involves the following steps:
1. Defining a local node
2. Adding a Token Ring Port
3. Defining a local LU
Using HP SNAplus2

The SNAplus2 main menu, from which you will start, is shown below:
Defining a local node

1. From the SNAplus2 main menu, select the Services pull-down:

   ![Configure node parameters](image)

   2. Select Configure node parameters... The following panel is displayed:

   ![X: Node parameters](image)

   3. Complete the Control point name with the values Network name (4) and Control point name (2).

   4. Enter the Control point name (2) in the Control point alias field.

   5. Enter the Node ID (3).

   6. Select End node.

   7. Press OK.

   A default independent local LU is defined.

Adding a Token Ring Port

1. From the main SNAplus2 menu, select the Connectivity and dependent LUs panel.

2. Press Add. The following panel is displayed:
3. Select a Token Ring Card port and press **OK**. The following panel is displayed:

![X: Add to lettuce](image1)

4. Enter the **SNA port name** (SF5800009/SF590000).
5. Enter a **Description** and press **OK** to take the default values.

**Defining a local LU**

1. From the main SNAplus2 menu, select the Independent local LUs panel.
2. Press **Add**. The following panel is displayed:
3. Enter the LU name (/SF5800005/SF590000) and press OK.

**APPC configuration**

APPC configuration involves the following steps:
1. Defining a remote node
2. Defining a partner LU
3. Defining a link station
4. Defining a mode
5. Adding CPI-C information
6. Adding a TP definition

**Defining a remote node**

1. From the main SNAplus2 menu, select the Remote systems panel.
2. Press Add. The following panel is displayed:

3. Select Define remote node and press OK. The following panel is displayed:
4. Enter the **Node’s SNA network name** (**OS2PU** 13) and a **Description**.
5. Press **OK**.
6. A default partner LU with the same name is generated and a message is displayed.
7. Press **OK**.

**Defining a partner LU**

1. From the main SNAplus2 menu, select the remote node in the Remote systems panel.
2. Press **Add**. The following panel is displayed:

3. Select **Define partner LU on node node name**.
4. Press **OK**. The following panel is displayed:
5. Enter the partner LU name (SF58000015/SF590000) and press OK.

Defining a link station

1. From the main SNAplus2 menu, select the Connectivity and dependent LUs panel.
2. Select the MQPORT port.
3. Press Add. The following panel is displayed:

4. Select Add link station to port MQPORT.
5. Press OK. The following panel is displayed:
6. Enter the **Name** of the link station (12).
7. Set the value of **Activation** to “On demand”.
8. Select **Independent only**.
9. Press **Remote node...** and select the value of the remote node (14).
10. Press **OK**.
11. Set the value of **Remote node type** to “End or LEN node”.
12. Enter the value for **MAC address** (19) and press **Advanced...**. The following panel is displayed:
13. Select **Reactivate link station after failure**.
14. Press **OK** to exit the Advanced... panel.
15. Press **OK** again.

**Defining a mode**

1. From the SNAplus2 main menu, select the **Services** pull-down: The following panel is displayed:

2. Select **APPC**. The following panel is displayed:
3. Select Modes.... The following panel is displayed:

4. Press Add. The following panel is displayed:
5. Enter the **Name** to be given to the mode (17).
6. Set the values of **Initial session limit** to 8, **Min con. winner sessions** to 4, and **Auto-activated sessions** to 0.
7. Press **OK**.
8. Press **Done**.

### Adding CPI-C information
1. From the SNAplus2 main menu, select the **Services** pull-down:

   ![Configure node parameters]

   - **Connectivity**
   - **3270**
   - **5250**
   - **RJE**
   - **LUA**
   - **APPC**
   - **IN3270 server**

2. Select **APPC**. The following panel is displayed:
Using HP SNAplus2

3. Select CPI-C... The following panel is displayed:

4. Press Add. The following panel is displayed:
5. Enter the Name (\[18\]). Select **Application TP** and enter the value (\[16\]). Select **Use PLU alias** and enter the name (\[15\]). Enter the **Mode** name (\[17\]).

6. Press **OK**.

**Adding a TP definition using HP SNAplus2 Release 5**

Invokable TP definitions are kept in the file `/etc/opt/sna/sna_tps`. This should be edited to add a TP definition. Add the following lines:

```
[MQSERIES]
PATH = /users/interops/HPUX.crs6a
TYPE = NON-QUEUED
USERID = mqm
ENV = APPCLU=HPUXLU
ENV = APPCTPN=MQSERIES
```

See “WebSphere MQ for HP-UX invokable TP setup” on page 241 for more information about TP definitions.

**Adding a TP definition using HP SNAplus2 Release 6**

To add a TP definition:

1. Select **Services** pulldown and select **APPC** as for CPI-C information.
2. Select **Transaction Programs**. The following panel is displayed:
3. Select **Add**. The following panel is displayed:

![Transaction Invocation Panel](image)

4. Enter **TP name** (7) in the **Application TP** field.
5. Mark **Incoming Allocates** as non-queued.
6. Enter **Full path to executable** (10).
7. Enter **-m Local queue manager** (11) in the **Arguments** field.
8. Enter **mqm** in the **User ID** and **Group ID** fields.
9. Enter environment variables **APPCLU=local LU** (5) and **APPCTPN=Invokable TP** (7) separated by the pipe character in the **Environment** field.
10. Press **OK** to save your definition.
**HP-UX operation**

The SNAplus2 control daemon is started with the `snap start` command. Depending on the options selected at installation, it may already be running.

The `xsnapadmin` utility controls SNAplus2 resources.

Logging and tracing are controlled from here. Log and trace files can be found in the `/var/opt/sna` directory. The logging files `sna.aud` and `sna.err` can be read using a standard editor such as `vi`.

In order to read the trace files `sna1.trc` and `sna2.trc` they must first be formatted by running the command `snaptcrfmt -f sna1.trc -o sna1` which produces a `sna1.dmp` file which can be read using a normal editor.

The configuration file itself is editable but this is not a recommended method of configuring SNAplus2.

The APPCLU environment variables must be set before starting a sender channel from the HP-UX system. The command can be either entered interactively or added to the logon profile. Depending on the level of BOURNE shell or KORN shell program being used, the command will be:

- `export APPCLU=HPUXLU /SF5800005/SF590000` newer level
- `export APPCLU=HPUXLU /older level`

**What next?**

The connection is now established. You are ready to complete the configuration. Go to "WebSphere MQ for HP-UX configuration" on page 238.

**Establishing a TCP connection**

1. Edit the file `/etc/services`.

   **Note:** To edit the `/etc/services` file, you must be logged in as a superuser or root. If you do not have the following line in that file, add it as shown:

   ```
   MQSeries 1414/tcp # MQSeries channel listener
   ```

2. Edit the file `/etc/inetd.conf`. If you do not have the following line in that file, add it as shown:

   ```
   MQSeries stream tcp nowait root /opt/mqm/bin/amqcrsta amqcrsta [-m queue.manager.name]
   ```

3. Find the process ID of the inetd with the command:

   ```
   ps -ef | grep inetd
   ```

4. Run the command:

   ```
   kill -1 processid
   ```

   **Note:** You must add `root` to the `mqm` group. You do not need not have the primary group set to `mqm`. As long as `mqm` is in the set of groups, you can use the commands. If you are running only applications that use the queue manager you do not need to have `mqm` group authority.
HP-UX and TCP

What next?

The connection is now established. You are ready to complete the configuration.
Go to "WebSphere MQ for HP-UX configuration"

WebSphere MQ for HP-UX configuration

Before beginning the installation process ensure that you have first created the 
mqm user and group, and set the password.

Start any channel using the command:
runmqchl -c channel.name

Notes:
1. Sample programs are installed in /opt/mqm/samp.
2. Error logs are stored in /var/mqm/qmgrs/qmgrname/errors.
3. When you are using the command interpreter runmqsc to enter administration 
commands, a + at the end of a line indicates that the next line is a continuation. 
Ensure that there is a space between the last parameter and the continuation 
character.

Basic configuration

1. Create the queue manager from the UNIX prompt using the command:
crtmqm -u dlqname -q hpux

where:
hpux Is the name of the queue manager
-q Indicates that this is to become the default queue manager
-u dlqname Specifies the name of the undeliverable message queue

This command creates a queue manager and a set of default objects. It sets the 
DEADQ attribute of the queue manager but does not create the undeliverable 
message queue.

2. Start the queue manager from the UNIX prompt using the command:
strmqm hpux

where hpux is the name given to the queue manager when it was created.

Channel configuration

The following section details the configuration to be performed on the HP-UX 
queue manager to implement the channel described in Figure 32 on page 101.

In each case the MQSC command is shown. Either start runmqsc from a UNIX 
prompt and enter each command in turn, or build the commands into a command 
file.

Examples are given for connecting WebSphere MQ for HP-UX and MQSeries for 
OS/2 Warp. If you wish connect to another WebSphere MQ product use the 
appropriate set of values from the table in place of those for OS/2.

Note: The words in bold are user-specified and reflect the names of WebSphere 
MQ objects used throughout these examples. If you change the names used
here, ensure that you also change the other references made to these objects throughout this book. All others are keywords and should be entered as shown.

Table 25. Configuration worksheet for WebSphere MQ for HP-UX

<table>
<thead>
<tr>
<th>ID</th>
<th>Parameter Name</th>
<th>Reference</th>
<th>Example Used</th>
<th>User Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>Queue Manager Name</td>
<td>HPUX</td>
<td></td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>Local queue name</td>
<td>HPUX.LOCALQ</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Definition for local node

Connection to MQSeries for OS/2 Warp

The values in this section of the table must match those used in Table 16 on page 161, as indicated.

<table>
<thead>
<tr>
<th>C</th>
<th>Remote queue manager name</th>
<th>A</th>
<th>OS2</th>
</tr>
</thead>
<tbody>
<tr>
<td>D</td>
<td>Remote queue name</td>
<td>OS2.REMOTEQ</td>
<td></td>
</tr>
<tr>
<td>E</td>
<td>Queue name at remote system</td>
<td>B</td>
<td>OS2.LOCALQ</td>
</tr>
<tr>
<td>F</td>
<td>Transmission queue name</td>
<td>OS2</td>
<td></td>
</tr>
<tr>
<td>G</td>
<td>Sender (SNA) channel name</td>
<td>HPUX.OS2.SNA</td>
<td></td>
</tr>
<tr>
<td>H</td>
<td>Sender (TCP/IP) channel name</td>
<td>HPUX.OS2.TCP</td>
<td></td>
</tr>
<tr>
<td>I</td>
<td>Receiver (SNA) channel name</td>
<td>C</td>
<td>OS2.HPUX.SNA</td>
</tr>
<tr>
<td>J</td>
<td>Receiver (TCP/IP) channel name</td>
<td>H</td>
<td>OS2.HPUX.TCP</td>
</tr>
</tbody>
</table>

Connection to WebSphere MQ for Windows

The values in this section of the table must match those used in Table 18 on page 184, as indicated.

<table>
<thead>
<tr>
<th>C</th>
<th>Remote queue manager name</th>
<th>A</th>
<th>WINNT</th>
</tr>
</thead>
<tbody>
<tr>
<td>D</td>
<td>Remote queue name</td>
<td>WINNT.REMOTEQ</td>
<td></td>
</tr>
<tr>
<td>E</td>
<td>Queue name at remote system</td>
<td>B</td>
<td>WINNT.LOCALQ</td>
</tr>
<tr>
<td>F</td>
<td>Transmission queue name</td>
<td>WINNT</td>
<td></td>
</tr>
<tr>
<td>G</td>
<td>Sender (SNA) channel name</td>
<td>HPUX.WINNT.SNA</td>
<td></td>
</tr>
<tr>
<td>H</td>
<td>Sender (TCP/IP) channel name</td>
<td>HPUX.WINNT.TCP</td>
<td></td>
</tr>
<tr>
<td>I</td>
<td>Receiver (SNA) channel name</td>
<td>C</td>
<td>WINNT.HPUX.SNA</td>
</tr>
<tr>
<td>J</td>
<td>Receiver (TCP/IP) channel name</td>
<td>H</td>
<td>WINNT.HPUX.TCP</td>
</tr>
</tbody>
</table>

Connection to WebSphere MQ for AIX

The values in this section of the table must match those used in Table 22 on page 210, as indicated.

<table>
<thead>
<tr>
<th>C</th>
<th>Remote queue manager name</th>
<th>A</th>
<th>AIX</th>
</tr>
</thead>
<tbody>
<tr>
<td>D</td>
<td>Remote queue name</td>
<td>AIX.REMOTEQ</td>
<td></td>
</tr>
<tr>
<td>E</td>
<td>Queue name at remote system</td>
<td>B</td>
<td>AIX.LOCALQ</td>
</tr>
<tr>
<td>F</td>
<td>Transmission queue name</td>
<td>AIX</td>
<td></td>
</tr>
<tr>
<td>G</td>
<td>Sender (SNA) channel name</td>
<td>HPUX.AIX.SNA</td>
<td></td>
</tr>
<tr>
<td>H</td>
<td>Sender (TCP) channel name</td>
<td>HPUX.AIX.TCP</td>
<td></td>
</tr>
<tr>
<td>I</td>
<td>Receiver (SNA) channel name</td>
<td>C</td>
<td>AIX.HPUX.SNA</td>
</tr>
<tr>
<td>J</td>
<td>Receiver (TCP) channel name</td>
<td>H</td>
<td>AIX.HPUX.TCP</td>
</tr>
</tbody>
</table>

Connection to MQSeries for Compaq Tru64 UNIX

The values in this section of the table must match those used in Table 23 on page 216, as indicated.

<table>
<thead>
<tr>
<th>C</th>
<th>Remote queue manager name</th>
<th>A</th>
<th>DECUX</th>
</tr>
</thead>
<tbody>
<tr>
<td>D</td>
<td>Remote queue name</td>
<td>DECUX.REMOTEQ</td>
<td></td>
</tr>
<tr>
<td>E</td>
<td>Queue name at remote system</td>
<td>B</td>
<td>DECUX.LOCALQ</td>
</tr>
<tr>
<td>F</td>
<td>Transmission queue name</td>
<td>DECUX</td>
<td></td>
</tr>
<tr>
<td>G</td>
<td>Sender (TCP) channel name</td>
<td>DECUX.HPUX.TCP</td>
<td></td>
</tr>
<tr>
<td>J</td>
<td>Receiver (TCP) channel name</td>
<td>H</td>
<td>HPUX.DECUX.TCP</td>
</tr>
</tbody>
</table>
## HP-UX configuration

Table 25. Configuration worksheet for WebSphere MQ for HP-UX (continued)

<table>
<thead>
<tr>
<th>ID</th>
<th>Parameter Name</th>
<th>Reference</th>
<th>Example Used</th>
<th>User Value</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><strong>Connection to WebSphere MQ for z/OS without CICS</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>Remote queue manager name</td>
<td>GIS</td>
<td></td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>Remote queue name</td>
<td>GIS.REMOTEQ</td>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>Queue name at remote system</td>
<td>GIS.LOCALQ</td>
<td></td>
<td></td>
</tr>
<tr>
<td>D</td>
<td>Transmission queue name</td>
<td>GIS</td>
<td></td>
<td></td>
</tr>
<tr>
<td>E</td>
<td>Sender (SNA) channel name</td>
<td>HPUX.GIS.SNA</td>
<td></td>
<td></td>
</tr>
<tr>
<td>F</td>
<td>Sender (TCP) channel name</td>
<td>HPUX.GIS.TCP</td>
<td></td>
<td></td>
</tr>
<tr>
<td>G</td>
<td>Receiver (SNA) channel name</td>
<td>GIS.HPUX.SNA</td>
<td></td>
<td></td>
</tr>
<tr>
<td>H</td>
<td>Receiver (TCP) channel name</td>
<td>GIS.HPUX.TCP</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Connection to WebSphere MQ for Solaris</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>Remote queue manager name</td>
<td>SOLARIS</td>
<td></td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>Remote queue name</td>
<td>SOLARIS.REMOTEQ</td>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>Queue name at remote system</td>
<td>SOLARIS.LOCALQ</td>
<td></td>
<td></td>
</tr>
<tr>
<td>D</td>
<td>Transmission queue name</td>
<td>SOLARIS</td>
<td></td>
<td></td>
</tr>
<tr>
<td>E</td>
<td>Sender (SNA) channel name</td>
<td>HPUX.SOLARIS.SNA</td>
<td></td>
<td></td>
</tr>
<tr>
<td>F</td>
<td>Sender (TCP/IP) channel name</td>
<td>HPUX.SOLARIS.TCP</td>
<td></td>
<td></td>
</tr>
<tr>
<td>G</td>
<td>Receiver (SNA) channel name</td>
<td>SOLARIS.HPUX.SNA</td>
<td></td>
<td></td>
</tr>
<tr>
<td>H</td>
<td>Receiver (TCP/IP) channel name</td>
<td>SOLARIS.HPUX.TCP</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Connection to WebSphere MQ for iSeries</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>Remote queue manager name</td>
<td>AS400</td>
<td></td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>Remote queue name</td>
<td>AS400.REMOTEQ</td>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>Queue name at remote system</td>
<td>AS400.LOCALQ</td>
<td></td>
<td></td>
</tr>
<tr>
<td>D</td>
<td>Transmission queue name</td>
<td>AS400</td>
<td></td>
<td></td>
</tr>
<tr>
<td>E</td>
<td>Sender (SNA) channel name</td>
<td>HPUX.AS400.SNA</td>
<td></td>
<td></td>
</tr>
<tr>
<td>F</td>
<td>Sender (TCP/IP) channel name</td>
<td>HPUX.AS400.TCP</td>
<td></td>
<td></td>
</tr>
<tr>
<td>G</td>
<td>Receiver (SNA) channel name</td>
<td>AS400.HPUX.SNA</td>
<td></td>
<td></td>
</tr>
<tr>
<td>H</td>
<td>Receiver (TCP) channel name</td>
<td>AS400.HPUX.TCP</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Connection to WebSphere MQ for z/OS without CICS</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>Remote queue manager name</td>
<td>MVS</td>
<td></td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>Remote queue name</td>
<td>MVS.REMOTEQ</td>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>Queue name at remote system</td>
<td>MVS.LOCALQ</td>
<td></td>
<td></td>
</tr>
<tr>
<td>D</td>
<td>Transmission queue name</td>
<td>MVS</td>
<td></td>
<td></td>
</tr>
<tr>
<td>E</td>
<td>Sender (SNA) channel name</td>
<td>HPUX.MVS.SNA</td>
<td></td>
<td></td>
</tr>
<tr>
<td>F</td>
<td>Sender (TCP) channel name</td>
<td>HPUX.MVS.TCP</td>
<td></td>
<td></td>
</tr>
<tr>
<td>G</td>
<td>Receiver (SNA) channel name</td>
<td>MVS.HPUX.SNA</td>
<td></td>
<td></td>
</tr>
<tr>
<td>H</td>
<td>Receiver (TCP) channel name</td>
<td>MVS.HPUX.TCP</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Connection to MQSeries for VSE/ESA</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>Remote queue manager name</td>
<td>VSE</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Table 25. Configuration worksheet for WebSphere MQ for HP-UX (continued)

<table>
<thead>
<tr>
<th>ID</th>
<th>Parameter Name</th>
<th>Reference</th>
<th>Example Used</th>
<th>User Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>D</td>
<td>Remote queue name</td>
<td>VSE.REMOTEQ</td>
<td></td>
<td></td>
</tr>
<tr>
<td>E</td>
<td>Queue name at remote system</td>
<td>VSE.LOCALQ</td>
<td></td>
<td></td>
</tr>
<tr>
<td>F</td>
<td>Transmission queue name</td>
<td>VSE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>G</td>
<td>Sender channel name</td>
<td>HPUX.VSE.SNA</td>
<td></td>
<td></td>
</tr>
<tr>
<td>I</td>
<td>Receiver channel name</td>
<td>VSE.HPUX.SNA</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**WebSphere MQ for HP-UX sender-channel definitions using SNA**

```bash
def q1 (OS2) +
    usage(xmitq) +
    replace

def qr (OS2.REMOTEQ) +
    rname(OS2.LOCALQ) +
    rqmname(OS2) +
    xmitq(OS2) +
    replace

def chl (HPUX.OS2.SNA) chltype(sdr) +
    trptype(lu62) +
    connname('OS2CPIC') +
    xmitq(OS2) +
    replace
```

**WebSphere MQ for HP-UX receiver-channel definitions using SNA**

```bash
def q1 (HPUX.LOCALQ) replace

def chl (OS2.HPUX.SNA) chltype(rcvr) +
    trptype(lu62) +
    replace
```

**WebSphere MQ for HP-UX invokable TP setup**

This is not required for HP SNAplus2 Release 6.

During the HP SNAplus2 configuration process, you created an invokable TP definition, which points to an executable file. In the example, the file was called /users/interops/HPUX.crs6a. You can choose what you call this file, but you are recommended to include the name of your queue manager in the name. The contents of the executable file must be:

```bash
#!/bin/sh
/opt/mqm/bin/amqcrs6a -m hpux
```

where `hpux` is the name of your queue manager.

This ensures that SNA receiver channels activate correctly when a sender channel initiates a conversation.

**WebSphere MQ for HP-UX sender-channel definitions using TCP**

```bash
def q1 (OS2) +
    usage(xmitq) +
    replace

def qr (OS2.REMOTEQ) +
    rname(OS2.LOCALQ) +
    rqmname(OS2) +
    xmitq(OS2) +
    replace
```
HP-UX configuration

```python
def chl (HPUX.OS2.TCP) chltype(sdr) +
    trptypet(tcp) +
    conname(remote_tcpip_hostname) +
    xmitq(OS2) +
    replace
```

WebSphere MQ for HP-UX receiver-channel definitions using TCP/IP

```python
def ql (HPUX.LOCALQ) replace
```

```python
def chl (OS2.HPUX.TCP) chltype(rcvr) +
    trptype(tcp) +
    replace
```
This chapter gives an example of how to set up communication links from MQSeries for AT&T GIS UNIX to WebSphere MQ products on the following platforms:
- OS/2
- Windows
- AIX
- Compaq Tru64 UNIX
- HP-UX
- Solaris
- OS/400
- z/OS without CICS
- VSE/ESA

First it describes the parameters needed for an LU 6.2 connection, then it describes "Establishing a connection using AT&T GIS SNA Server" on page 246 and "Establishing a TCP connection" on page 251.

Once the connection is established, you need to define some channels to complete the configuration. This is described in "Channel configuration" on page 252.

See Chapter 7, “Example configuration chapters in this book” on page 101 for background information about this chapter and how to use it.

### Configuration parameters for an LU 6.2 connection

Table 26 presents a worksheet listing all the parameters needed to set up communication from AT&T GIS UNIX to one of the other WebSphere MQ platforms. The worksheet shows examples of the parameters, which have been tested in a working environment, and leaves space for you to fill in your own values. An explanation of the parameter names follows the worksheet. Use the worksheet in this chapter in conjunction with the worksheet in the chapter for the platform to which you are connecting.

### Configuration worksheet

Use the following worksheet to record the values you will use for this configuration. Where numbers appear in the Reference column they indicate that the value must match that in the appropriate worksheet elsewhere in this book. The examples that follow in this chapter refer back to the values in the ID column of this table. The entries in the Parameter Name column are explained in "Explanation of terms" on page 246.

<table>
<thead>
<tr>
<th>ID</th>
<th>Parameter Name</th>
<th>Reference</th>
<th>Example</th>
<th>User Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Configuration</td>
<td></td>
<td>010</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Network name</td>
<td></td>
<td>NETID</td>
<td></td>
</tr>
</tbody>
</table>

5. This platform has become NCR UNIX SVR4 MP-RAS, R3.0

© Copyright IBM Corp. 1993, 2002
## AT&T GIS UNIX and LU 6.2

### Table 26. Configuration worksheet for AT&T GIS SNA Services (continued)

<table>
<thead>
<tr>
<th>ID</th>
<th>Parameter Name</th>
<th>Reference</th>
<th>Example</th>
<th>User Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>Control Point name</td>
<td>GISPU</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Local LU name</td>
<td>GISLU</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>LU 6.2 Transaction Program name</td>
<td>MQSERIES</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>Local PU name</td>
<td>GISPU</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>Mode name</td>
<td>#INTER</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>Token-Ring adapter address</td>
<td>10007038E86B</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>Local XID</td>
<td>03E 00018</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Connection to an OS/2 system**

The values in this section of the table must match those used in Table 15 on page 144, as indicated.

| 10 | Remote Node name                        | OS2PU     |                  |            |
| 11 | Network name                            | NETID     |                  |            |
| 12 | Remote LU name                          | OS2LU     |                  |            |
| 13 | Remote Transaction Program name         | MQSERIES  |                  |            |
| 14 | LU 6.2 CPI-C side information symbolic destination | OS2CPIC |                  |            |
| 15 | Mode name                               | #INTER    |                  |            |
| 16 | LAN destination address                 | 10005AFC5D83 |              |            |

**Connection to a Windows system**

The values in this section of the table must match those used in Table 17 on page 168, as indicated.

| 10 | Remote Node name                        | WINNTCP   |                  |            |
| 11 | Network name                            | NETID     |                  |            |
| 12 | Remote LU name                          | WINNTLU   |                  |            |
| 13 | Remote Transaction Program name         | MQSERIES  |                  |            |
| 14 | LU 6.2 CPI-C side information symbolic destination | NTCPIC |                  |            |
| 15 | Mode name                               | #INTER    |                  |            |
| 16 | LAN destination address                 | 08005AA5FAB9 |              |            |

**Connection to an AIX system**

The values in this section of the table must match those used in Table 21 on page 197, as indicated.

| 10 | Remote Node name                        | AIXPU     |                  |            |
| 11 | Network name                            | NETID     |                  |            |
| 12 | Remote LU name                          | AIXLU     |                  |            |
| 13 | Remote Transaction Program name         | MQSERIES  |                  |            |
| 14 | LU 6.2 CPI-C side information symbolic destination | AIXCPIC |                  |            |
| 15 | Mode name                               | #INTER    |                  |            |
| 16 | LAN destination address                 | 123456789012 |              |            |

**Connection to an HP-UX system**

The values in this section of the table must match those used in Table 24 on page 219, as indicated.

| 10 | Remote Node name                        | HPUXPU    |                  |            |
| 11 | Network name                            | NETID     |                  |            |
| 12 | Remote LU name                          | HPUXLU    |                  |            |
| 13 | Remote Transaction Program name         | MQSERIES  |                  |            |
| 14 | LU 6.2 CPI-C side information symbolic destination | HPUXCPIC |                  |            |
### Table 26. Configuration worksheet for AT&T GIS SNA Services (continued)

<table>
<thead>
<tr>
<th>ID</th>
<th>Parameter Name</th>
<th>Reference</th>
<th>Example</th>
<th>User Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>Mode name</td>
<td>6</td>
<td>#INTER</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>LAN destination address</td>
<td>8</td>
<td>100090DC2C7C</td>
<td></td>
</tr>
</tbody>
</table>

**Connection to a Solaris system**

The values in this section of the table must match those used in Table 28 on page 258, as indicated.

<table>
<thead>
<tr>
<th>ID</th>
<th>Parameter Name</th>
<th>Reference</th>
<th>Example</th>
<th>User Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>Remote Node name</td>
<td>3</td>
<td>SOLARPU</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>Network name</td>
<td>2</td>
<td>NETID</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>Remote LU name</td>
<td>7</td>
<td>SOLARLU</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>Remote Transaction Program name</td>
<td>8</td>
<td>MQSERIES</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>LU 6.2 CPI-C side information symbolic destination</td>
<td></td>
<td>SOLCPIC</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>Mode name</td>
<td>17</td>
<td>#INTER</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>LAN destination address</td>
<td>5</td>
<td>08002071CC8A</td>
<td></td>
</tr>
</tbody>
</table>

**Connection to an iSeries system**

The values in this section of the table must match those used in Table 48 on page 521, as indicated.

<table>
<thead>
<tr>
<th>ID</th>
<th>Parameter Name</th>
<th>Reference</th>
<th>Example</th>
<th>User Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>Remote Node name</td>
<td>2</td>
<td>AS400PU</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>Network name</td>
<td>1</td>
<td>NETID</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>Remote LU name</td>
<td>3</td>
<td>AS400LU</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>Remote Transaction Program name</td>
<td>8</td>
<td>MQSERIES</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>LU 6.2 CPI-C side information symbolic destination</td>
<td></td>
<td>AS4CPIC</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>Mode name</td>
<td>17</td>
<td>#INTER</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>LAN destination address</td>
<td>4</td>
<td>10005A5962EF</td>
<td></td>
</tr>
</tbody>
</table>

**Connection to a z/OS system without CICS**

The values in this section of the table must match those used in Table 33 on page 366, as indicated.

<table>
<thead>
<tr>
<th>ID</th>
<th>Parameter Name</th>
<th>Reference</th>
<th>Example</th>
<th>User Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>Remote Node name</td>
<td>3</td>
<td>MVSPU</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>Network name</td>
<td>2</td>
<td>NETID</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>Remote LU name</td>
<td>4</td>
<td>MVSLU</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>Remote Transaction Program name</td>
<td>7</td>
<td>MQSERIES</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>LU 6.2 CPI-C side information symbolic destination</td>
<td></td>
<td>MVSCPIC</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>Mode name</td>
<td>10</td>
<td>#INTER</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>LAN destination address</td>
<td>8</td>
<td>400074511092</td>
<td></td>
</tr>
</tbody>
</table>

**Connection to a VSE/ESA system**

The values in this section of the table must match those used in Table 50 on page 549, as indicated.

<table>
<thead>
<tr>
<th>ID</th>
<th>Parameter Name</th>
<th>Reference</th>
<th>Example</th>
<th>User Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>Remote Node name</td>
<td>2</td>
<td>VSEPU</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>Network name</td>
<td>1</td>
<td>NETID</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>Remote LU name</td>
<td>3</td>
<td>VSELU</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>Remote Transaction Program name</td>
<td>4</td>
<td>MQ01</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>LU 6.2 CPI-C side information symbolic destination</td>
<td></td>
<td>VSECPCIC</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>Mode name</td>
<td></td>
<td>#INTER</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>LAN destination address</td>
<td>5</td>
<td>400074511092</td>
<td></td>
</tr>
</tbody>
</table>
Explanation of terms

1. **Configuration**
   This is the unique ID of the SNA Server configuration you are creating or modifying. Valid values are between 0 and 255.

2. **Network name**
   This is the unique ID of the network to which you are connected. Your network administrator will tell you this value.

3. **Control Point name**
   This is a unique Control Point name for this workstation. Your network administrator will assign this to you.

4. **Local LU name**
   A logical unit (LU) manages the exchange of data between systems. The local LU name is the name of the LU on your system. Your network administrator will assign this to you.

5. **LU 6.2 Transaction Program name**
   WebSphere MQ applications trying to converse with this workstation will specify a symbolic name for the program to be run at the receiving end. This will have been defined on the channel definition at the sender. Wherever possible we use a transaction program name of MQSERIES, or in the case of a connection to VSE/ESA, where the length is limited to 4 bytes, use MQTF.

   See Table 20 on page 194 for more information.

6. **Local PU name**
   This is a unique PU name for this workstation. Your network administrator will assign this to you.

7. **Mode name**
   This is the name given to the set of parameters that control the APPC conversation. This name must be defined at each partner system. Your network administrator will assign this to you.

8. **Token-ring adapter address**
   The is the 12-character hex address of the token-ring card.

9. **Remote Node name**
   This is a meaningful symbolic name by which the connection to a partner node is known. It is used only inside SNA Server setup and is specified by you.

10. **LU 6.2 CPI-C Side Information Symbolic Destination**
    This is a name given to the definition of a partner node. You supply the name. It need be unique only on this machine. You will later use the name in the WebSphere MQ sender channel definition.

**Establishing a connection using AT&T GIS SNA Server**

The following information guides you through the tasks you must perform to create the SNA infrastructure that WebSphere MQ requires. This example creates the definitions for a new partner node and LU on OS/2.

Use `snamgr` to enter the AT&T GIS SNA Server configuration panels. You need root authority to use `snamgr`.

246 WebSphere MQ Intercommunication
Throughout the following example, only the panels containing information that must be added or updated are shown. Preceding each panel is a list of the sequence of panels that you must invoke to proceed from the initial menu to the relevant customization panel.

Note: SNA Server works better in an Xterm session than it does in an ASCII session such as TELNET.

Defining local node characteristics
Setting up the local node involves the following steps:
1. Configuring the SNA subsystem
2. Defining a mode
3. Defining a local Transaction Program

Configuring the SNA subsystem
Proceed through these panels:

1 SNA Manager
2 Configuration
   3 SNA Subsystem Configuration
   4 SNA Subsystem Configuration Creation

The panels are shown as figures representing the screens that appear. Instructions for completing the screens and entering the correct information to configure the SNA subsystem are given in the text after each figure.

Enter the configuration identifier (/SF5800001/SF590000).

Enter Y.

Enter the configuration identifier (1).

Enter Y.

1 SNA Configuration of the Local Node
Node Parameters:
   Node ID of Local Node 00
   PU Resource Name (optional) GISPU
   Network Identifier (optional) NETID
   Control Point (CP) Name (optional) GISPU
Local LU 6.2 Parameters:
   LU 6.2 Logical Unit Name GISLU
   Max Number of LU 6.2 Sessions 0100
Using AT&T GIS SNA Server

Enter the values for Node ID of Local Node, PU Resource Name (6), Network Identifier (2), CP Name (3), LU 6.2 Logical Unit Name (4), and Max Number of LU 6.2 Sessions.

Defining a mode
Proceed through these panels:
2 Local Configuration

Select Define a mode.

```
3 Conversation Mode Definition

Mode Name                       #INTER
Maximum Number of Sessions     008
Number of Locally Controlled Sessions 004
Honor Pending Conversation Requests Before an Existing Session is Terminated? N
Number of Automatically Established Sessions 004
Code Set to be Used During Transmission of TP Data E
```

Enter the values for Mode Name (7), Maximum Number of Sessions, and Number of Locally Controlled Sessions.

```
4 Conversation Mode Definition for Max RU

Send Max RU Size Upper Bound    03840
Send Max RU Size Lower Bound    00128
Receive Max RU Size Upper Bound 03840
Receive Max RU Size Lower Bound 00128
```

Defining a local Transaction Program
2 Local Configuration

Select Define a RECEIVE_ALLOCATE local TP.

```
3 Receive_Allocate Transaction Program Definition

TP name                MQSERIES_______________________
TP start type          A (M = Manual, A = Automatic)
receive_allocate timer (seconds) -1__ (0 - 9999, -1)
Incoming allocate timer (seconds) -1__ (0 - 9999, -1)
Max number of auto-started TP instances 1_ (1 - 99)
```

Enter the values for TP name (5), and set the TP start type to A.
Note: Before this will work you need to associate the TP name with an executable program. You do this outside `snamgr` by creating a symbolic link entry in the directory `/usr/lbin` either before or after you configure SNA Server. Enter the following commands:

```
  cd /usr/lbin
  ln -s /opt/mqm/bin/amqcrs6a MQSeries
```

Connecting to a partner node

To connect to a partner node you need to:
1. Configure a remote node
2. Define a partner LU
3. Add a CPI-C Side Entry

Configuring a remote node

Proceed through these panels:

2 Local Configuration

Select **End Local Configuration**.

1 Remote Node Definition

Select **Peer Node Definition**.

<table>
<thead>
<tr>
<th>2 Remote Node Configuration</th>
</tr>
</thead>
<tbody>
<tr>
<td>Remote Node Name</td>
</tr>
<tr>
<td>Type of Link Connection</td>
</tr>
<tr>
<td>SNA Logical Connection ID</td>
</tr>
<tr>
<td>Link to Backup (Optional)</td>
</tr>
</tbody>
</table>

Enter the values for **Remote Node Name**, **Type of Link Connection**, and **SNA Logical Connection ID**.

<table>
<thead>
<tr>
<th>3 SNA/TR Configuration for Connection 01</th>
</tr>
</thead>
<tbody>
<tr>
<td>Token Ring Adapter ID</td>
</tr>
<tr>
<td>Maximum Send BTU Length</td>
</tr>
<tr>
<td>Local XID</td>
</tr>
<tr>
<td>Data link role of local system</td>
</tr>
<tr>
<td>Remote DLSAP</td>
</tr>
<tr>
<td>Remote MAC Address</td>
</tr>
<tr>
<td>Route Discovery Command</td>
</tr>
<tr>
<td>Broadcast Timer</td>
</tr>
</tbody>
</table>

Enter the values for **Token Ring Adapter ID**, **Local XID**, and **Remote MAC address**.
Defining a partner LU
Proceed through these panels:

1 LU 6.2 Logical Unit Definition
To complete the definition of Remote Peer Node, OS2, you need to define at least one Remote LU 6.2 Logical Unit.
Press CONT to Continue.

2 Partner LU 6.2 Definition
Locally Known Name OS2LU
Network Identifier NETID
Network Name (LUNAME) OS2LU
Uninterpreted Name OS2LU
Session Capability P

Enter the values for Locally Known Name (12), Network Identifier (11), Network Name (LUNAME) (12), and Uninterpreted Name (12).

3 Automatic Activation
Auto Activate at Start of Day? N

4 LU 6.2 Partner Definition
Do you want to define another remote LU 6.2 Logical Unit in the remote node, OS2? N

Adding a CPI-C Side Entry
Proceed through these panels:
1 SNA MANAGER
2 Configuration
   3 CPI-C Side Information
Enter the name of the CPI-C Side Information File (14).

Enter the values for Partner LU name (12), Mode name (15), and TP name (13).

What next?
The LU 6.2 connection is now established. You are ready to complete the configuration. Go to “MQSeries for AT&T GIS UNIX configuration” on page 252.

Establishing a TCP connection

1. Edit the file /etc/services.

   Note: To edit the /etc/services file, you must be logged in as a superuser or root. If you do not have the following line in that file, add it as shown:
   MQSeries 1414/tcp # MQSeries channel listener

2. Edit the file /usr/etc/inetd.conf. If you do not have the following line in that file, add it as shown:
   MQSeries stream tcp nowait root /opt/mqm/bin/amqcrsta amqcrsta
   [-m queue.manager.name]

3. Find the process ID of the inetd with the command:
   ps -ef | grep inetd

4. Run the command:
   kill -1 inetd processid

   The command kill -1 can be unreliable. If it doesn’t work, use the command kill -9 and then restart /usr/etc/inetd manually.

What next?
The LU 6.2 connection is now established. You are ready to complete the configuration. Go to “MQSeries for AT&T GIS UNIX configuration” on page 252.
Before beginning the installation process ensure that you have first created the *mqm* user and group, and set the password.

Start any channel using the command:
```
runmqchl -c channel.name
```

**Notes:**

1. Sample programs are installed in `/opt/mqm/samp`.
2. Error logs are stored in `/var/mqm/qmgrs/qmgrname/errors`.
3. When you are using the command interpreter `runmqsc` to enter administration commands, a `+` at the end of a line indicates that the next line is a continuation. Ensure that there is a space between the last parameter and the continuation character.

**Basic configuration**

1. Create the queue manager from the UNIX prompt using the command:
   ```
crtmqm -u dlqname -q gis
   ```
   where:
   - `gis` Is the name of the queue manager
   - `-q` Indicates that this is to become the default queue manager
   - `-u dlqname` Specifies the name of the undeliverable message queue

2. Start the queue manager from the UNIX prompt using the command:
   ```
   strmqm gis
   ```
   where `gis` is the name given to the queue manager when it was created.

3. Before creating your own objects you must first create the system default objects. These are a number of definitions for required objects and templates on which user definitions will be modelled.

   Create the default objects from the UNIX prompt using the command:
   ```
   runmqsc gis < /opt/mqm/samp/amqscoma.tst > defobj.out
   ```
   where `gis` is the name of the queue manager. Display the file `defobj.out` and ensure that all objects were created successfully. There is a report at the end of the file.

**Channel configuration**

The following section details the configuration to be performed on the AT&T GIS UNIX queue manager to implement the channel described in Figure 32 on page 101.

In each case the MQSC command is shown. Either start `runmqsc` from a UNIX prompt and enter each command in turn, or build a command file of the same format as `amqscoma.tst` and use it as before to create the objects.

Examples are given for connecting MQSeries for AT&T GIS UNIX and MQSeries for OS/2 Warp. If you wish to connect to another WebSphere MQ product use the appropriate set of values from the table in place of those for OS/2.
### AT&T GIS UNIX configuration

**Note:** The words in **bold** are user-specified and reflect the names of WebSphere MQ objects used throughout these examples. If you change the names used here, ensure that you also change the other references made to these objects throughout this book. All others are keywords and should be entered as shown.

Table 27. Configuration worksheet for MQSeries for AT&T GIS UNIX

<table>
<thead>
<tr>
<th>ID</th>
<th>Parameter Name</th>
<th>Reference</th>
<th>Example Used</th>
<th>User Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>Queue Manager Name</td>
<td>GIS</td>
<td></td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>Local queue name</td>
<td>GIS.LOCALQ</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Definition for local node**

**Connection to MQSeries for OS/2 Warp**

The values in this section of the table must match those used in Table 16 on page 161 as indicated.

| C  | Remote queue manager name    | A          | OS2          |            |
| D  | Remote queue name            |           | OS2.REMOTEQ  |            |
| E  | Queue name at remote system  | B          | OS2.LOCALQ   |            |
| F  | Transmission queue name      |            | OS2          |            |
| G  | Sender (SNA) channel name    |            | GIS.OS2.SNA  |            |
| H  | Sender (TCP/IP) channel name |            | GIS.OS2.TCP  |            |
| I  | Receiver (SNA) channel name  | G          | OS2.GIS.SNA  |            |
| J  | Receiver (TCP/IP) channel name|          | OS2.GIS.TCP  |            |

**Connection to WebSphere MQ for Windows**

The values in this section of the table must match those used in Table 18 on page 184 as indicated.

| C  | Remote queue manager name    | A          | WINNT        |            |
| D  | Remote queue name            |           | WINNT.REMOTEQ|            |
| E  | Queue name at remote system  | B          | WINNT.LOCALQ |            |
| F  | Transmission queue name      |            | WINNT        |            |
| G  | Sender (SNA) channel name    |            | GIS.WINNT.SNA|            |
| H  | Sender (TCP/IP) channel name |            | GIS.WINNT.TCP|            |
| I  | Receiver (SNA) channel name  | G          | WINNT.GIS.SNA|            |
| J  | Receiver (TCP/IP) channel name|        | WINNT.GIS.TCP|            |

**Connection to WebSphere MQ for AIX**

The values in this section of the table must match those used in Table 22 on page 210 as indicated.

| C  | Remote queue manager name    | A          | AIX          |            |
| D  | Remote queue name            |           | AIX.REMOTEQ  |            |
| E  | Queue name at remote system  | B          | AIX.LOCALQ   |            |
| F  | Transmission queue name      |            | AIX          |            |
| G  | Sender (SNA) channel name    |            | GIS.AIX.SNA  |            |
| H  | Sender (TCP) channel name    |            | GIS.AIX.TCP  |            |
| I  | Receiver (SNA) channel name  | G          | AIX.GIS.SNA  |            |
| J  | Receiver (TCP) channel name  | H          | AIX.GIS.TCP  |            |

**Connection to MQSeries for Compaq Tru64 UNIX**

The values in this section of the table must match those used in Table 23 on page 216 as indicated.

| C  | Remote queue manager name    | A          | DECUX        |            |
| D  | Remote queue name            |           | DECUX.REMOTEQ|            |
| E  | Queue name at remote system  | B          | DECUX.LOCALQ |            |
| F  | Transmission queue name      |            | DECUX        |            |
### AT&T GIS UNIX configuration

**Table 27. Configuration worksheet for MQSeries for AT&T GIS UNIX (continued)**

<table>
<thead>
<tr>
<th>ID</th>
<th>Parameter Name</th>
<th>Reference</th>
<th>Example Used</th>
<th>User Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>Sender (TCP) channel name</td>
<td></td>
<td>DECUX.GIS.TCP</td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>Receiver (TCP) channel name</td>
<td></td>
<td>GIS.DECUX.TCP</td>
<td></td>
</tr>
</tbody>
</table>

**Connection to WebSphere MQ for HP-UX**

The values in this section of the table must match those used in Table 25 on page 239, as indicated.

| C  | Remote queue manager name       | A                | HPUX             |                  |
| D  | Remote queue name               |                  | HPUX.REMOTEQ     |                  |
| E  | Queue name at remote system     | B                | HPUX.LOCALQ      |                  |
| F  | Transmission queue name         |                  | HPUX             |                  |
| G  | Sender (SNA) channel name       |                  | GIS.HPUX.SNA     |                  |
| H  | Sender (TCP) channel name       |                  | GIS.HPUX.TCP     |                  |
| I  | Receiver (SNA) channel name     | C                | HPUX.GIS.SNA     |                  |
| J  | Receiver (TCP) channel name     |                  | HPUX.GIS.TCP     |                  |

**Connection to WebSphere MQ for Solaris**

The values in this section of the table must match those used in Table 30 on page 290, as indicated.

| C  | Remote queue manager name       |                  | SOLARIS          |                  |
| D  | Remote queue name               |                  | SOLARIS.REMOTEQ  |                  |
| E  | Queue name at remote system     | B                | SOLARIS.LOCALQ   |                  |
| F  | Transmission queue name         |                  | SOLARIS          |                  |
| G  | Sender (SNA) channel name       |                  | GIS.SOLARIS.SNA  |                  |
| H  | Sender (TCP/IP) channel name    |                  | GIS.SOLARIS.TCP  |                  |
| I  | Receiver (SNA) channel name     | C                | SOLARIS.GIS.SNA  |                  |
| J  | Receiver (TCP/IP) channel name  |                  | SOLARIS.GIS.TCP  |                  |

**Connection to WebSphere MQ for iSeries**

The values in this section of the table must match those used in Table 49 on page 535, as indicated.

| C  | Remote queue manager name       |                  | AS400            |                  |
| D  | Remote queue name               |                  | AS400.REMOTEQ    |                  |
| E  | Queue name at remote system     | B                | AS400.LOCALQ     |                  |
| F  | Transmission queue name         |                  | AS400            |                  |
| G  | Sender (SNA) channel name       |                  | GIS.AS400.SNA    |                  |
| H  | Sender (TCP/IP) channel name    |                  | GIS.AS400.TCP    |                  |
| I  | Receiver (SNA) channel name     | C                | AS400.GIS.SNA    |                  |
| J  | Receiver (TCP) channel name     |                  | AS400.GIS.TCP    |                  |

**Connection to WebSphere MQ for z/OS without CICS**

The values in this section of the table must match those used in Table 34 on page 372, as indicated.

| C  | Remote queue manager name       |                  | MVS              |                  |
| D  | Remote queue name               |                  | MVS.REMOTEQ      |                  |
| E  | Queue name at remote system     | B                | MVS.LOCALQ       |                  |
| F  | Transmission queue name         |                  | MVS              |                  |
| G  | Sender (SNA) channel name       |                  | GIS.MVS.SNA      |                  |
| H  | Sender (TCP) channel name       |                  | GIS.MVS.TCP      |                  |
| I  | Receiver (SNA) channel name     | C                | MVS.GIS.SNA      |                  |
| J  | Receiver (TCP) channel name     |                  | MVS.GIS.TCP      |                  |
Table 27. Configuration worksheet for MQSeries for AT&T GIS UNIX (continued)

<table>
<thead>
<tr>
<th>ID</th>
<th>Parameter Name</th>
<th>Reference</th>
<th>Example Used</th>
<th>User Value</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Connection to MQSeries for VSE/ESA</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>The values in this section of the table must match those used in Table 51 on page 554 as indicated.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>Remote queue manager name</td>
<td>VSE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>D</td>
<td>Remote queue name</td>
<td>VSE.REMOTEQ</td>
<td></td>
<td></td>
</tr>
<tr>
<td>E</td>
<td>Queue name at remote system</td>
<td>VSE.LOCALQ</td>
<td></td>
<td></td>
</tr>
<tr>
<td>F</td>
<td>Transmission queue name</td>
<td>VSE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>G</td>
<td>Sender channel name</td>
<td>GIS.VSE.SNA</td>
<td></td>
<td></td>
</tr>
<tr>
<td>I</td>
<td>Receiver channel name</td>
<td>VSE.GIS.SNA</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

MQSeries for AT&T GIS UNIX sender-channel definitions using SNA

```bash
def ql (OS2) +
  usage(xmitq) +
  replace

def qr (OS2.REMOTEQ) +
  rname(OS2.LOCALQ) +
  rqmname(OS2) +
  xmitq(OS2) +
  replace

def chl (GIS.OS2.SNA) chltype(sdr) +
  trptype(lu62) +
  conname('OS2CPIC') +
  xmitq(OS2) +
  replace
```

MQSeries for AT&T GIS UNIX receiver-channel definitions using SNA

```bash
def ql (GIS.LOCALQ) replace

def chl (OS2.GIS.SNA) chltype(rcvr) +
  trptype(lu62) +
  replace
```

MQSeries for AT&T GIS UNIX sender-channel definitions using TCP

```bash
def ql (OS2) +
  usage(xmitq) +
  replace

def qr (OS2.REMOTEQ) +
  rname(OS2.LOCALQ) +
  rqmname(OS2) +
  xmitq(OS2) +
  replace

def chl (GIS.OS2.TCP) chltype(sdr) +
  trptype(tcp) +
  conname(remote_tcpip_hostname) +
  xmitq(OS2) +
  replace
```
MQSeries for AT&T GIS UNIX receiver-channel definitions using TCP/IP

def ql (GIS.LOCALQ) replace

def chl (OS2.GIS.TCP) chltype(rcvr) +
    trptype(tcp) +
    replace
Chapter 18. Example configuration - IBM WebSphere MQ for Solaris

This chapter gives an example of how to set up communication links from WebSphere MQ for Solaris to WebSphere MQ products on the following platforms:

- OS/2
- Windows
- AIX
- Compaq Tru64 UNIX
- HP-UX
- AT&T GIS UNIX 6
- OS/400
- z/OS without CICS
- VSE/ESA

It describes the parameters needed for an LU 6.2 connection, using SunLink Version 9.1, and “Establishing a connection using SunLink Version 9.1” on page 262.

WebSphere MQ allows you to set up communication links from WebSphere MQ for Solaris using SNAP-IX V6.2 or later. See “Configuration parameters for an LU 6.2 connection using SNAP-IX” on page 272 and “Establishing a session using SNAP-IX” on page 276.

To establish a TCP connection, see “Establishing a TCP connection” on page 288.

Once the connection is established, you need to define some channels to complete the configuration. This is described in “WebSphere MQ for Solaris configuration” on page 289.

See Chapter 7, “Example configuration chapters in this book” on page 101 for background information about this chapter and how to use it.

Loading the WebSphere MQ library to support SNA

You determine the WebSphere MQ library loaded to support SNA with the MQCommLibrary parameter in the qm.ini file. On WebSphere MQ the LU62 stanza applies. MQCommLibrary is the only attribute that can be specified:

MQCommLibrary=amqcc62a | amqcc62s

This attribute specifies the WebSphere MQ library loaded to support SNA for WebSphere MQ V5.3.

SNAP-IX support is provided if amqcc62a is loaded. This is the default if this attribute is not specified.

SunLink Version 9.1 support is provided if amqcc62s is loaded.

Refer to the WebSphere MQ System Administration Guide book for a full description of the LU62 stanza in the qm.ini file.

6. This platform has become NCR UNIX SVR4 MP-RAS, R3.0

© Copyright IBM Corp. 1993, 2002
WebSphere MQ for Solaris

If you are migrating from previous versions of MQSeries other than MQSeries V5.2, to use SunLink 9.1, you must correctly specify the LU62 stanza entry to be used for SNA communication. If you do not specify the LU62 stanza entry correctly, the following errors are returned:

- When attempting to start an outbound LU62 channel, message AMQ6175 is returned, and the following entry appears in the queue manager error log:
  
  AMQ6175: The APPC communications program could not be loaded

  Explanation:
  The Attempt to load the APPC library or procedure 'amqcc62a' failed with error code 536895861

- When attempting to start an inbound LU62 channel, the following errors are produced on the sending machine:

  AMQ9201: Allocate failed to host 'TEST'
  AMQ9999: Channel program ended abnormally

  See WebSphere MQ Messages for details of AMQ9201, the contents of which are produced in the sending queue manager error log.

Configuration parameters for an LU 6.2 connection using SunLink Version 9.1

Table 28 presents a worksheet listing all the parameters needed to set up communication from Solaris to one of the other WebSphere MQ platforms, using SunLink Version 9.1. The worksheet shows examples of the parameters, which have been tested in a working environment, and leaves space for you to fill in your own values. An explanation of the parameter names follows the worksheet. Use the worksheet in this chapter in conjunction with the worksheet in the chapter for the platform to which you are connecting.

### Configuration worksheet

Use this worksheet to record the values you use for your configuration. Where numbers appear in the Reference column they indicate that the value must match that in the appropriate worksheet elsewhere in this book. The examples that follow in this chapter refer back to the values in the ID column. The entries in the Parameter Name column are explained in "Explanation of terms" on page 261.

#### Table 28. Configuration worksheet for SunLink Version 9.1

<table>
<thead>
<tr>
<th>ID</th>
<th>Parameter Name</th>
<th>Reference</th>
<th>Example</th>
<th>User Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>PU 2.1 server name</td>
<td>SOLSERV</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Network name</td>
<td>NETID</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>CP name</td>
<td>SOLARPU</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Line name</td>
<td>MQLINE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Local MAC address</td>
<td>08002071CC8A</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>Local terminal ID</td>
<td>05D310D6</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>Local LU name</td>
<td>SOLARLU</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>TP name</td>
<td>MQSERIES</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>Command Path</td>
<td>home/interops/crs6a</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Parameters for local node**

**Connection to an OS/2 system**

The values in this section of the table must match those used in Table 15 on page 144, as indicated.

| 10 | Unique session name | OS2SESS |
### Table 28. Configuration worksheet for SunLink Version 9.1 (continued)

<table>
<thead>
<tr>
<th>ID</th>
<th>Parameter Name</th>
<th>Reference</th>
<th>Example</th>
<th>User Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>11</td>
<td>Network name</td>
<td>2</td>
<td>NETID</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>DLC name</td>
<td>2</td>
<td>OS2QMGR</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>Remote CP name</td>
<td>2</td>
<td>OS2PU</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>Local LSAP</td>
<td>2</td>
<td>x'04', x'08', x'0C', ...</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>Partner LU</td>
<td>6</td>
<td>OS2LU</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>TP name</td>
<td>8</td>
<td>MQSERIES</td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>Mode name</td>
<td>2</td>
<td>#INTER</td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>CPI-C file name</td>
<td>10</td>
<td>/home/mqstart/OS2CPIC</td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>Remote MAC address</td>
<td>10</td>
<td>10005AFCD83</td>
<td></td>
</tr>
</tbody>
</table>

**Connection to a Windows system**

The values in this section of the table must match those used in Table 17 on page 168, as indicated.

<table>
<thead>
<tr>
<th>ID</th>
<th>Parameter Name</th>
<th>Reference</th>
<th>Example</th>
<th>User Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>Unique session name</td>
<td>10</td>
<td>WINNTSESS</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>Network name</td>
<td>2</td>
<td>NETID</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>DLC name</td>
<td>2</td>
<td>NTQMGR</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>Remote CP name</td>
<td>2</td>
<td>WINNTPU</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>Local LSAP</td>
<td>2</td>
<td>x'04', x'08', x'0C', ...</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>Partner LU</td>
<td>6</td>
<td>WINNTLU</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>TP name</td>
<td>8</td>
<td>MQSERIES</td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>Mode name</td>
<td>2</td>
<td>#INTER</td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>CPI-C file name</td>
<td>10</td>
<td>/home/mqstart/NTCPIC</td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>Remote MAC address</td>
<td>10</td>
<td>10005AFCD83</td>
<td></td>
</tr>
</tbody>
</table>

**Connection to an AIX system**

The values in this section of the table must match those used in Table 21 on page 197, as indicated.

<table>
<thead>
<tr>
<th>ID</th>
<th>Parameter Name</th>
<th>Reference</th>
<th>Example</th>
<th>User Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>Unique session name</td>
<td>10</td>
<td>AIXSESS</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>Network name</td>
<td>1</td>
<td>NETID</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>DLC name</td>
<td>1</td>
<td>AIXQMGR</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>Remote CP name</td>
<td>2</td>
<td>AIXPU</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>Local LSAP</td>
<td>2</td>
<td>x'04', x'08', x'0C', ...</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>Partner LU</td>
<td>4</td>
<td>AIXLU</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>TP name</td>
<td>6</td>
<td>MQSERIES</td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>Mode name</td>
<td>14</td>
<td>#INTER</td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>CPI-C file name</td>
<td>10</td>
<td>/home/mqstart/AIXCPIC</td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>Remote MAC address</td>
<td>15</td>
<td>10005AFCD83</td>
<td></td>
</tr>
</tbody>
</table>

**Connection to an HP-UX system**

The values in this section of the table must match those used in Table 24 on page 219, as indicated.

<table>
<thead>
<tr>
<th>ID</th>
<th>Parameter Name</th>
<th>Reference</th>
<th>Example</th>
<th>User Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>Unique session name</td>
<td>10</td>
<td>HPUXSESS</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>Network name</td>
<td>4</td>
<td>NETID</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>DLC name</td>
<td>4</td>
<td>HPUXQMGR</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>Remote CP name</td>
<td>5</td>
<td>HPUXPU</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>Local LSAP</td>
<td>5</td>
<td>x'04', x'08', x'0C', ...</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>Partner LU</td>
<td>7</td>
<td>HPUXLU</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>TP name</td>
<td>7</td>
<td>MQSERIES</td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>Mode name</td>
<td>17</td>
<td>#INTER</td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>CPI-C file name</td>
<td>10</td>
<td>/home/mqstart/HPCPIC</td>
<td></td>
</tr>
</tbody>
</table>
### Table 28. Configuration worksheet for SunLink Version 9.1 (continued)

<table>
<thead>
<tr>
<th>ID</th>
<th>Parameter Name</th>
<th>Reference</th>
<th>Example</th>
<th>User Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>19</td>
<td>Remote MAC address</td>
<td>19</td>
<td>10005AFC5D83</td>
<td></td>
</tr>
</tbody>
</table>

#### Connection to an AT&T GIS UNIX system

The values in this section of the table must match those used in [Table 26 on page 243](#) as indicated.

<table>
<thead>
<tr>
<th>10</th>
<th>Unique session name</th>
<th>GISSESS</th>
</tr>
</thead>
<tbody>
<tr>
<td>11</td>
<td>Network name</td>
<td>NETID</td>
</tr>
<tr>
<td>12</td>
<td>DLC name</td>
<td>GISQMGR</td>
</tr>
<tr>
<td>13</td>
<td>Remote CP name</td>
<td>GISPU</td>
</tr>
<tr>
<td>14</td>
<td>Local LSAP</td>
<td>x'04', x'08', x'0C', ...</td>
</tr>
<tr>
<td>15</td>
<td>Partner LU</td>
<td>GISLU</td>
</tr>
<tr>
<td>16</td>
<td>TP name</td>
<td>MQSERIES</td>
</tr>
<tr>
<td>17</td>
<td>Mode name</td>
<td>#INTER</td>
</tr>
<tr>
<td>18</td>
<td>CPI-C file name</td>
<td>/home/mqstart/ATTCPIC</td>
</tr>
<tr>
<td>19</td>
<td>Remote MAC address</td>
<td>10005AFC5D83</td>
</tr>
</tbody>
</table>

#### Connection to an iSeries system

The values in this section of the table must match those used in [Table 48 on page 521](#) as indicated.

<table>
<thead>
<tr>
<th>10</th>
<th>Unique session name</th>
<th>AS400SESS</th>
</tr>
</thead>
<tbody>
<tr>
<td>11</td>
<td>Network name</td>
<td>NETID</td>
</tr>
<tr>
<td>12</td>
<td>DLC name</td>
<td>ASQMGR</td>
</tr>
<tr>
<td>13</td>
<td>Remote CP name</td>
<td>AS400PU</td>
</tr>
<tr>
<td>14</td>
<td>Local LSAP</td>
<td>x'04', x'08', x'0C', ...</td>
</tr>
<tr>
<td>15</td>
<td>Partner LU</td>
<td>AS400LU</td>
</tr>
<tr>
<td>16</td>
<td>TP name</td>
<td>MQSERIES</td>
</tr>
<tr>
<td>17</td>
<td>Mode name</td>
<td>#INTER</td>
</tr>
<tr>
<td>18</td>
<td>CPI-C file name</td>
<td>/home/mqstart/400CPIC</td>
</tr>
<tr>
<td>19</td>
<td>Remote MAC address</td>
<td>10005AFC5D83</td>
</tr>
</tbody>
</table>

#### Connection to a z/OS system without CICS

The values in this section of the table must match those used in [Table 33 on page 366](#) as indicated.

<table>
<thead>
<tr>
<th>10</th>
<th>Unique session name</th>
<th>MVSSESS</th>
</tr>
</thead>
<tbody>
<tr>
<td>11</td>
<td>Network name</td>
<td>NETID</td>
</tr>
<tr>
<td>12</td>
<td>DLC name</td>
<td>MVSQMGR</td>
</tr>
<tr>
<td>13</td>
<td>Remote CP name</td>
<td>MVSPU</td>
</tr>
<tr>
<td>14</td>
<td>Local LSAP</td>
<td>x'04', x'08', x'0C', ...</td>
</tr>
<tr>
<td>15</td>
<td>Partner LU</td>
<td>MVSLU</td>
</tr>
<tr>
<td>16</td>
<td>TP name</td>
<td>MQSERIES</td>
</tr>
<tr>
<td>17</td>
<td>Mode name</td>
<td>#INTER</td>
</tr>
<tr>
<td>18</td>
<td>CPI-C file name</td>
<td>/home/mqstart/MVSCPIC</td>
</tr>
<tr>
<td>19</td>
<td>Remote MAC address</td>
<td>10005AFC5D83</td>
</tr>
</tbody>
</table>

#### Connection to a VSE/ESA system

The values in this section of the table must match those used in [Table 50 on page 549](#) as indicated.

<table>
<thead>
<tr>
<th>10</th>
<th>Unique session name</th>
<th>VSESESS</th>
</tr>
</thead>
<tbody>
<tr>
<td>11</td>
<td>Network name</td>
<td>NETID</td>
</tr>
<tr>
<td>12</td>
<td>DLC name</td>
<td>VSEQMGR</td>
</tr>
<tr>
<td>13</td>
<td>Remote CP name</td>
<td>VSEPU</td>
</tr>
<tr>
<td>14</td>
<td>Local LSAP</td>
<td>x'04', x'08', x'0C', ...</td>
</tr>
</tbody>
</table>
### Table 28. Configuration worksheet for SunLink Version 9.1 (continued)

<table>
<thead>
<tr>
<th>ID</th>
<th>Parameter Name</th>
<th>Reference</th>
<th>Example</th>
<th>User Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>Partner LU</td>
<td></td>
<td>VSELU</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>TP name</td>
<td></td>
<td>MQSERIES</td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>Mode name</td>
<td></td>
<td>#INTER</td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>CPI-C file name</td>
<td></td>
<td>/home/mqstart/VSECPIC</td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>Remote MAC address</td>
<td></td>
<td>10005AFC5D83</td>
<td></td>
</tr>
</tbody>
</table>

### Explanation of terms

1. **PU2.1 server name**
   This is the name of the PU2.1 server for the local control point.

2. **Network name**
   This is the unique ID of the network to which you are connected. It is an alphanumeric value and can be 1-8 characters long. The network name works with the Control Point name to uniquely identify a system. Your network administrator will tell you the value.

3. **CP name**
   This is the unique Control Point name for this workstation. Your network administrator will assign this to you.

4. **Line name**
   This is the name that identifies the connection to the LAN.

5. **Local MAC address**
   This is the network address of the token-ring card. The address to be specified is found in the ether value displayed in response to the `ifconfig tr0` command issued at a root level of authority. (Tr0 is the name of the machine’s token-ring interface.) If you do not have the necessary level of authority, your Solaris system administrator can tell you the value.

6. **Local terminal ID**
   This is the unique ID of this workstation. On other platforms this is often referred to as the *Exchange ID* or *XID*. Your network administrator will assign this ID for you.

7. **Local LU name**
   An LU manages the exchange of data between transactions. The local LU name is the name of the LU on your system. Your network administrator will assign this to you.

8. **TP name**
   WebSphere MQ applications trying to converse with this workstation will specify a symbolic name for the program to be run at the receiving end. This will have been defined on the channel definition at the sender. For simplicity, wherever possible use a transaction program name of MQSERIES, or in the case of a connection to VSE/ESA, where the length is limited to 4 bytes, use MQ01.

   See [Table 20 on page 194](#) for more information.

9. **TP path**
   This is the path and name of the script file that invokes the WebSphere MQ program to run.

10. **Unique session name**
    This is the unique name of the Partner LU/Mode definition.
Solaris and LU 6.2

12 DLC name
This is the name of the link to the remote system.

13 Remote CP name
This is the name of the control point on the remote system.

18 CPI-C file name
This is the full path and name of the file which holds CPI-C side information for a partner system. There must be a separate CPI-C file for each partner. For increased flexibility, include the full path and file name in the WebSphere MQ sender channel definition.

Establishing a connection using SunLink Version 9.1

This section describes how to establish a connection using SunLink Version 9.1. The topics discussed are:

- SunLink 9.0 base configuration
- Invokable TPs
- CPI-C side information

SunLink 9.1 base configuration

To start the SunLink 9.1 graphical interface:

1. Enter `sungmi` at the command line.
   
   It is assumed that the domain, manager systems, and default system were defined during installation.

2. On the main screen, highlight Config1 in the resource tree and select File and Open. A window entitled Connect to domain appears:

   ![Connect to domain window](image)

3. Enter required details to connect to the required domain.

Configuring a PU 2.1 server

1. Double click on Systems in the resource tree to display a list of systems.

2. Double click on System name in the resource tree to open its subordinate entries.

3. Using the right mouse button, highlight PU2.1 Servers in the resource tree and select New and PU2.1 Server from the pop-up menu. A window entitled Create PU2.1 Server appears:
4. Enter the PU2.1 Name (1).
5. Enter the CP Name. This consists of the Network Name (2) and the CP Name (3).
6. Click on Advanced >>. The advanced window appears:

7. Enter the SunOp Service and LU6.2 Service
8. Click on OK when you are happy with the settings.

**Adding a LAN connection**

1. Double click on PU2.1 Servers in the resource tree to display the name of the PU2.1 server.
Using SunLink

2. Using the right mouse button, highlight the server name in the resource tree and select New and LAN Connection from the pop-up menu. A window entitled Create LAN Connection appears:

```
| Line Name:  | MOLINE |
| Comment:    |        |

Device Information

Type:    | Token-Ring (302,5) |
Device:  | /dev/trg:          |

Addressing

Local MAC Address: 08002371CC8A
```

3. Enter a Line Name (4) and Local MAC Address (5).
4. Click on Advanced>> The advanced window appears:
5. Check the **LAN Speed** is correct.
6. Click on **OK** when you are happy with the settings.

**Configuring a connection to a remote PU**

1. Double click the **PU2.1 server name** in the resource tree to open its subordinate entries.
2. Double click on **LAN Connections**.
3. Using the right mouse button, highlight the LAN connection name in the resource tree and select **New** and **DLC** from the pop-up menu. A window entitled **Create DLC** appears:
4. Enter the DLC Name (SF58000012/SF590000) and Remote MAC Address (SF58000019/SF590000).

5. Click on Advanced>>. A window entitled Create DLC (advanced) appears:

6. Enter the Local LSAP for this DLC (SF58000014/SF590000), Local Terminal ID (SF5800006/SF590000), and Remote CP Name (SF58000013/SF590000).

7. When you are happy with the settings, click on OK.

**Configuring an independent LU**

1. Double click on Systems in the resource tree to display a list of systems.
2. Double click on the system name to open its subordinate entries.
3. Double click on PU2.1 Servers to display a list of servers.
4. Double click on the PU2.1 server name to open its subordinate entries.
5. From the main window, select Edit, New, and Independent LU to display the Create Independent LU window:

![Create Independent LU window](image)

6. Enter the Local LU Name (7).

7. Click on Advanced>>. An advanced Create Independent LU window appears:
8. Enter the **Network Qual Name**. This consists of the **Network Name** and the **Local LU**.

9. Click on **OK**

**Configuring a partner LU**

1. Double click on the PU2.1 server name in the resource tree to open its subordinate entries.
2. From the main window, select **Edit**, **New**, and **Partner LU** to display the **Create Partner LU** window:
3. Enter the Partner LU (15) and the Local LU Name (7).
4. Click on Advanced>>. The advanced Create Partner LU window appears:

5. Choose a Local LU from the drop-down list.
6. Click on OK.

**Configuring the session mode**

1. Double click on the PU2.1 server name to open its subordinate entries.
2. Double click on Partner LU in the resource tree to display a list of partner LUs.
3. Click on the partner LU to select it.
4. From the main window, select Edit, New, and Mode to display the Create Mode window:
5. Enter the **Mode Name** (SF58000017/SF590000) and **DLC Name** (SF58000012/SF590000).
6. Click on Advanced>>. The advanced Create Mode window appears:

7. Enter the **Unique Session Name** (SF58000010/SF590000).
8. When you are happy with the settings, click on **OK**.

**Configuring a transaction program**

1. Double click on the PU2.1 server name to open its subordinate entries.
2. Click on **Transaction Programs** in the resource tree to select it.
3. From the main window, select **Edit**, **New**, and **Transaction Program** to display the Create Transaction Program window:
4. Enter the TP Name (8) and Local LU (7).
5. Enter a path to the invokable TP in the Command Path (9) field:
6. Click on Advanced>>. The advanced Create Transaction Program window appears:

7. When you are happy with the settings, click on OK.

**Invokable TP path**
In order to set required environment variables a script file should be defined for each invokable TP containing the following:

```
#!/bin/ksh
export APPC_GATEWAY=zinfandel
export APPC_LOCAL_LU=SOLARLU
/opt/mqm/bin/amqcrs6a -m SOLARIS -n MQSERIES
```

**CPI-C side information**
In common with most other platforms, WebSphere MQ for Solaris uses CPI-C side information files (18) to hold information about its partner systems. In SunLink 9.1, these are ASCII files (one per partner).
Using SunLink

The location of the file must be specified either explicitly in the conname parameter of the sender channel definition or in the search path. It is better to specify it fully in the conname parameter because the value of the PATH environment variable can vary from user to user.

What next?

The connection is now established. You are ready to complete the configuration. Go to “WebSphere MQ for Solaris configuration” on page 289.

Configuration parameters for an LU 6.2 connection using SNAP-IX

Table 29 presents a worksheet listing all the parameters needed to set up communication from Solaris using SNAP-IX to one of the other WebSphere MQ platforms. The worksheet shows examples of the parameters, which have been tested in a working environment, and leaves space for you to fill in your own values. An explanation of the parameter names follows the worksheet. Use the worksheet for the platform to which you are connecting.

Configuration worksheet

Use this worksheet to record the values you use for your configuration. Where numbers appear in the Reference column they indicate that the value must match that in the appropriate worksheet elsewhere in this book. The examples that follow in this chapter refer to the values in the ID column. The entries in the Parameter Name column are explained in “Explanation of terms” on page 274.

Table 29. Configuration worksheet for SNAP-IX

<table>
<thead>
<tr>
<th>ID</th>
<th>Parameter Name</th>
<th>Ref.</th>
<th>Example</th>
<th>User Value</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Parameters for local node</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>Configuration file name</td>
<td></td>
<td>sna_node.cfg</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Control point name</td>
<td></td>
<td>SOLARXPU</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Node ID to send</td>
<td></td>
<td>05D 23456</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Network name</td>
<td></td>
<td>NETID</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Local APPC LU</td>
<td></td>
<td>SOLARXLU</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>APPC mode</td>
<td></td>
<td>#INTER</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>Invokable TP</td>
<td></td>
<td>MQSERIES</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>Local MAC address</td>
<td></td>
<td>08002071CC8A</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>Port name</td>
<td></td>
<td>MQPORT</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>Command path</td>
<td></td>
<td>/opt/mqm/bin/amqcrs6a</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>Local queue manager</td>
<td></td>
<td>SOLARIS</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Connection to an OS/2 system</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>Link station name</td>
<td></td>
<td>OS2CONN</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>Network name</td>
<td></td>
<td>2 NETID</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>CP name</td>
<td></td>
<td>3 OS2PU</td>
<td></td>
</tr>
</tbody>
</table>

Figure 35. CPI-C side information file for SunLink Version 9.0

The values in this section of the table must match those used in the Table for OS/2 and LU6.2, as indicated.
Table 29. Configuration worksheet for SNAP-IX (continued)

<table>
<thead>
<tr>
<th>ID</th>
<th>Parameter Name</th>
<th>Ref.</th>
<th>Example</th>
<th>User Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>Remote LU</td>
<td>8</td>
<td>OS2LU</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>Application TP</td>
<td>8</td>
<td>MQSERIES</td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>Mode name</td>
<td>17</td>
<td>#INTER</td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>CPI-C symbolic destination name</td>
<td>342</td>
<td>OS2CPIC</td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>Remote network address</td>
<td>10</td>
<td>10005AFC5D83</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>Node ID to receive</td>
<td>4</td>
<td>05D 12345</td>
<td></td>
</tr>
</tbody>
</table>

**Connection to a Windows system**

The values in this section of the table must match those used in the Table for Windows and LU6.2, as indicated.

<table>
<thead>
<tr>
<th>ID</th>
<th>Parameter Name</th>
<th>Ref.</th>
<th>Example</th>
<th>User Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>12</td>
<td>Link station name</td>
<td>2</td>
<td>NETID</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>Network name</td>
<td>3</td>
<td>WINNTCP</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>CP name</td>
<td>4</td>
<td>WINNTLU</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>Remote LU</td>
<td>5</td>
<td>MQSERIES</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>Mode name</td>
<td>17</td>
<td>#INTER</td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>CPI-C symbolic destination name</td>
<td>342</td>
<td>NTCPIC</td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>Remote network address</td>
<td>9</td>
<td>08005AA5FAB9</td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>Node ID to receive</td>
<td>4</td>
<td>05D 30F65</td>
<td></td>
</tr>
</tbody>
</table>

**Connection to an AIX system**

The values in this section of the table must match those used in the Table for AIX and LU6.2, as indicated.

<table>
<thead>
<tr>
<th>ID</th>
<th>Parameter Name</th>
<th>Ref.</th>
<th>Example</th>
<th>User Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>12</td>
<td>Link station name</td>
<td>1</td>
<td>AIXCONN</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>Network name</td>
<td>2</td>
<td>NETID</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>CP name</td>
<td>3</td>
<td>AIXPU</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>Remote LU</td>
<td>4</td>
<td>AIXLU</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>Application TP</td>
<td>5</td>
<td>MQSERIES</td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>Mode name</td>
<td>17</td>
<td>#INTER</td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>CPI-C symbolic destination name</td>
<td>342</td>
<td>AIXCPIC</td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>Remote network address</td>
<td>8</td>
<td>123456789012</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>Node ID to receive</td>
<td>3</td>
<td>071 23456</td>
<td></td>
</tr>
</tbody>
</table>

**Connection to an AT&T GIS (NCR) UNIX System**

The values in this section of the table must match those used in the table the Table for AT&T GIS UNIX and LU6.2, as indicated.

<table>
<thead>
<tr>
<th>ID</th>
<th>Parameter Name</th>
<th>Ref.</th>
<th>Example</th>
<th>User Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>12</td>
<td>Link station name</td>
<td>2</td>
<td>GISCONN</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>Network name</td>
<td>3</td>
<td>NETID</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>CP name</td>
<td>4</td>
<td>GISPU</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>Remote LU</td>
<td>5</td>
<td>GISLU</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>Application TP</td>
<td>6</td>
<td>MQSERIES</td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>Mode name</td>
<td>7</td>
<td>#INTER</td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>CPI-C symbolic destination name</td>
<td>342</td>
<td>GISCPIC</td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>Remote network address</td>
<td>8</td>
<td>10007038E86B</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>Node ID to receive</td>
<td>3</td>
<td>03E 00018</td>
<td></td>
</tr>
</tbody>
</table>

**Connection to an HP—UX system**

The values in this section of the table must match those used in the Table for HP—UX and LU6.2, as indicated.

<table>
<thead>
<tr>
<th>ID</th>
<th>Parameter Name</th>
<th>Ref.</th>
<th>Example</th>
<th>User Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>12</td>
<td>Link station name</td>
<td>2</td>
<td>HPUXCONN</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>Network name</td>
<td>3</td>
<td>NETID</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>CP name</td>
<td>4</td>
<td>HPUXPU</td>
<td></td>
</tr>
</tbody>
</table>
### Sun Solaris and LU 6.2

**Table 29. Configuration worksheet for SNAP-IX (continued)**

<table>
<thead>
<tr>
<th>ID</th>
<th>Parameter Name</th>
<th>Ref.</th>
<th>Example</th>
<th>User Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>Remote LU</td>
<td>7</td>
<td>HPUXLU</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>Application TP</td>
<td>8</td>
<td>MQSERIES</td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>Mode name</td>
<td>17</td>
<td>#INTER</td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>CPI-C symbolic destination name</td>
<td></td>
<td>HPUXCPIC</td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>Remote network address</td>
<td>5</td>
<td>10005FC5D83</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>node ID to receive</td>
<td>6</td>
<td>05D 54321</td>
<td></td>
</tr>
</tbody>
</table>

**Connection to an iSeries system**

The values in this section of the table must match those used in the Table for AS/400® and LU6.2, as indicated.

<table>
<thead>
<tr>
<th>ID</th>
<th>Parameter Name</th>
<th>Ref.</th>
<th>Example</th>
<th>User Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>12</td>
<td>Link station name</td>
<td>1</td>
<td>NETID</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>Network name</td>
<td>2</td>
<td>AS400PU</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>CP name</td>
<td>3</td>
<td>AS400LU</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>Remote LU</td>
<td>4</td>
<td>MVSLU</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>Application TP</td>
<td>7</td>
<td>MQSERIES</td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>Mode name</td>
<td>10</td>
<td>#INTER</td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>CPI-C symbolic destination name</td>
<td></td>
<td>MVSCPIC</td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>Remote network address</td>
<td>8</td>
<td>400074511092</td>
<td></td>
</tr>
</tbody>
</table>

**Connection to a z/OS system without CICS**

The values in this section of the table must match those used in the Table for z/OS and LU6.2, as indicated.

<table>
<thead>
<tr>
<th>ID</th>
<th>Parameter Name</th>
<th>Ref.</th>
<th>Example</th>
<th>User Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>12</td>
<td>Link station name</td>
<td>2</td>
<td>MVSCONN</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>Network name</td>
<td>3</td>
<td>MVSPU</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>CP name</td>
<td>4</td>
<td>MVSLU</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>Remote LU</td>
<td>7</td>
<td>MQSERIES</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>CPI-C symbolic destination name</td>
<td></td>
<td>MVSCPIC</td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>Remote network address</td>
<td>8</td>
<td>400074511092</td>
<td></td>
</tr>
</tbody>
</table>

**Connection to a VSE/ESA system**

The values in this section of the table must match those used in the Table for VSE/ESA and LU6.2, as indicated.

<table>
<thead>
<tr>
<th>ID</th>
<th>Parameter Name</th>
<th>Ref.</th>
<th>Example</th>
<th>User Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>12</td>
<td>Link station name</td>
<td>1</td>
<td>VSECONN</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>Network name</td>
<td>2</td>
<td>VSEPU</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>CP name</td>
<td>3</td>
<td>VSELU</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>Remote LU</td>
<td>4</td>
<td>MQ01</td>
<td>MQ01</td>
</tr>
<tr>
<td>16</td>
<td>CPI-C symbolic destination name</td>
<td></td>
<td>VSECPI</td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>Remote network address</td>
<td>5</td>
<td>400074511092</td>
<td></td>
</tr>
</tbody>
</table>

**Explanation of terms**

1. **Configuration file name**
   This is the unique name of the SNAP-IX configuration file. The default for this name is **sna_node.cfg**.

   Although it is possible to edit this file, it is strongly recommended that configuration is done using **xsnadmin**.
**Control point name**
This is the unique Control point name for this workstation. In the SNA network, the Control point is an addressable location (PU type 2.1). Your network administrator will assign this to you.

**Node ID to send**
This is the unique ID of this workstation. On other platforms this is often referred to as the *Exchange ID* or *XID*. Your network administrator will assign this ID for you.

**Network name**
This is the unique ID of the network to which you are connected. It is an alphanumeric value and can be 1-8 characters long. The network name works with the Control point name to uniquely identify a system. Your network administrator will tell you the value.

**Local APPC LU**
An LU manages the exchange of data between transactions. The local APPC LU name is the name of the LU on your system. Your network administrator will assign this to you.

**APPC mode**
This is the name given to the set of parameters that control the APPC conversation. This name must be defined at each partner system. Your network administrator will assign this to you.

**Invokable TP**
WebSphere MQ applications trying to converse with this workstation will specify a symbolic name for the program to be run at the receiving end. This will have been defined on the channel definition at the sender. For simplicity, wherever possible use a transaction program name of MQSERIES, or in the case of a connection to VSE/ESA, where the length is limited to 4 bytes, use MQTP.

**Local MAC address**
This is the network address of the token-ring card. The address to be specified is found in the ether value displayed in response to the `ifconfig tr0` command issued at a root level of authority. (Tr0 is the name of the machine’s token-ring interface.) If you do not have the necessary level of authority, your Sun Solaris system administrator can tell you the value.

**Port name**
This is a meaningful symbolic name that is used to associate the definitions with a network interface (in this case, a Token-Ring adapter). A separate Port must be defined for each physical device attached to the workstation.

**Full path to executable**
This is the path and name of the script file that invokes the WebSphere MQ program to run.

**Local queue manager**
This is the name of the queue manager on your local system.

**Link station name**
This is a meaningful symbolic name by which the connection to a peer or host node is known. It defines a logical path to the remote system. Its name is used only inside SNAP-IX and is specified by you. The connection must be associated with an existing Link and owned by one local node. You must define one connection for each partner or host system.
Sun Solaris and LU 6.2

18 CPI-C symbolic destination name
This is a name given to the definition of a partner node. You choose the name. It need be unique only on this machine. Later you can use this name in the WebSphere MQ sender channel definition.

20 Node ID to receive
This is the unique ID of the partner workstation with which you will be communicating. On other platforms this is often referred to as the Exchange ID or XID. For a connection to a host system any values except 000 FFFFF and FFF FFFFF may be specified. Your network administrator will assign this ID for you.

Establishing a session using SNAP-IX

The following information guides you through the tasks you must perform to create the SNA infrastructure that WebSphere MQ requires. This example creates the definitions for a partner node and LU on OS/2.

Use sna start followed by xsnaadmin to type the SNAP-IX configuration panels. You need root authority to use xsnaadmin.

SNAP-IX configuration

SNAP-IX configuration involves the following steps:
1. Defining a local node
2. Adding a Token Ring Port
3. Defining a local LU

The SNAP-IX main menu, from which you start, is shown here:
Defining a local node

1. From the SNAP-IX main menu, click the Services pull-down. The Services pull-down appears:

2. Click Configure node parameters. The following panel is displayed:

3. Complete the Control point name with the values Network name (1) and Control point name (2).
4. Type the Control point name (2) in the Control point alias field.
5. Type the Node ID (3).
6. Click End node.
7. Click OK.

A default independent local LU is defined.
Adding a Token Ring Port

1. From the main SNAP-IX menu, click Connectivity and dependent LUs.
2. Click Add. The following panel is displayed:

3. Click Token Ring Card and click OK. The following panel is displayed:

4. Type the SNA port name (SF5800009/SF590000).
5. Type a Description and click OK to take the default values.
Defining a local LU
1. From the main SNAP-IX menu, click Independent local LUs.
2. Click Add. The following panel is displayed:

3. Type the LU name (SF5800005/SF590000) and click OK.

APPC configuration
APPC configuration involves the following steps:
1. Defining a remote node
2. Defining a partner LU
3. Defining a link station
4. Defining a mode
5. Adding CPI-C information
6. Adding a TP definition

Defining a remote node
1. From the main SNAP-IX menu, click Remote systems.
2. Click Add. The following panel is displayed:

3. Select the Define remote node check box and click OK. The following panel is displayed:

4. Type the Node’s SNA network name (HEIDIB) and a Description.
Using SNAP-IX

5. Click OK.

6. A default partner LU with the same name is generated and a message is displayed.

7. Click OK.

Defining a partner LU

1. From the main SNAP-IX menu, click Remote systems and click the remote node.

2. Click Add. The following panel is displayed:

3. Select the Define partner LU on node node name check box.

4. Click OK. The following panel is displayed:

5. Type the partner LU name (15) and click OK.

Defining a link station

1. From the main SNAP-IX menu, click Connectivity and dependent LUs.

2. Click the MQPORT port.

3. Click Add. The following panel is displayed:
4. Select the **Add link station to port MQPORT** check box.
5. Click **OK**. The following panel is displayed:

![Token ring link station panel](image)

6. Type the **Name** of the link station (12).
7. Set the value of **Activation** to “On demand”.
8. Select the **Independent only** check box.
9. Click **Remote node** and select the value of the remote node (14).
10. Click **OK**.
11. Set the value of **Remote node type** to “End or LEN node”.
12. Type the value for **MAC address** (19) and click **Advanced**. The following panel is displayed:
13. Select the Request CP-CP sessions check box.
14. Select the Reactivate link station after failure check box.
15. Click OK to exit the Advanced panel.
16. Click OK again.

**Defining a mode**

1. From the SNAP-IX main menu, click the Services pull-down: The following panel is displayed:

2. Click APPC. The following panel is displayed:
3. Click **Modes**. The following panel is displayed:

![Modes Panel](image)

4. Click **Add**. The following panel is displayed:

![Mode Panel](image)

5. Type the **Name** to be given to the mode (SF58000017/SF590000).
6. Set the values of **Initial session limit** to 8, **Min con. winner sessions** to 4, and **Auto-activated sessions** to 0.
7. Click **OK**.
8. Click **Done**.

**Adding CPI-C information**

1. From the SNAP-IX main menu, click the **Services** pull-down:
2. Click APPC. The following panel is displayed:

3. Click CPI-C. The following panel is displayed:

4. Click Add. The following panel is displayed:
5. Type the Name (18). Select the Application TP check box and type the value (16). Select the Use PLU alias check box and type the name (15). Type the Mode name (17).

6. Click OK.
Adding a TP definition using SNAP-IX Release 6

To add a TP definition:

1. Click the Services pull-down and click APPC as for CPI-C information.
2. Click Transaction Programs. The following panel is displayed:

3. Click Add. The following panel is displayed:

4. Type TP name (7) in the Application TP field.
5. Clear the Queue incoming Allocates check box.
6. Type Full path to executable (10).
7. Type -m Local queue manager (11) in the Arguments field.
8. Type `mqm` in the User ID and Group ID fields.

9. Type environment variables `APPCLLU=localhost LU (5)` and `APPCTPN=Invokable TP (7)` separated by the pipe character in the Environment field.

10. Click OK to save your definition.
Using SNAP-IX

SNAP-IX operation

The SNAP-IX control daemon is started with the `sna start` command. Depending on the options selected at installation, it may already be running.

The `xsnaadmin` utility controls SNAP-IX resources.

Logging and tracing are controlled from here. Log and trace files can be found in the `/var/opt/sna` directory. The logging files `sna.aud` and `sna.err` can be read using a standard editor such as `vi`.

In order to read the trace files `sna1.trc` and `sna2.trc` you must first format them by running the command `snatrcfmt -f sna1.trc -o sna1`. This produces a `sna1.dmp` file that can be read using a normal editor.

It is possible to edit the configuration file, but this is not a recommended method of configuring SNAP-IX.

The `APPCLLU` environment variables must be set before starting a sender channel from the Solaris system. The command can be either entered interactively or added to the logon profile. Depending on the level of BOURNE shell or KORN shell program being used, the command will be:

```
export APPCLLU=SOLARXLU
```

or

```
APPCLLU=SOLARXLU
```

What next?

The connection is now established. You are ready to complete the configuration. Go to "WebSphere MQ for Solaris configuration" on page 289.

Establishing a TCP connection

To establish a TCP connection, follow these steps.

1. Edit the file `/etc/services`.

   **Note:** To edit the `/etc/services` file, you must be logged in as a superuser or root. If you do not have the following line in that file, add it as shown:

   ```
   MQSeries 1414/tcp # MQSeries channel listener
   ```

2. Edit the file `/etc/inetd.conf`. If you do not have the following line in that file, add it as shown:

   ```
   MQSeries stream tcp nowait mqm /opt/mqm/bin/amqcrsta amqcrsta [-m queue.manager.name]
   ```

3. Find the process ID of the inetd with the command:

   ```
   ps -ef | grep inetd
   ```

4. Run the command:

   ```
   kill -1 processid
   ```

   **Note:** You must add `root` to the `mqm` group. You do not need have the primary group set to `mqm`. As long as `mqm` is in the set of groups, you can use the commands. If you are running only applications that use the queue manager you do not need to have `mqm` group authority.
What next?

The TCP/IP connection is now established. You are ready to complete the configuration. Go to "WebSphere MQ for Solaris configuration".

WebSphere MQ for Solaris configuration

Before beginning the installation process ensure that you have first created the mqm user and group, and set the password.

Start any channel using the command:

```
runmqchl -c channel.name
```

Notes:
1. Sample programs are installed in /opt/mqm/samp.
2. Error logs are stored in /var/mqm/qmgrs/qmgrname/errors.
3. When you are using the command interpreter runmqsc to enter administration commands, a + at the end of a line indicates that the next line is a continuation. Ensure that there is a space between the last parameter and the continuation character.
4. For an SNA or LU6.2 channel, if you experience an error when you try to load the communications library, probably file liblu62.so cannot be found. A likely solution to this problem is to add its location, which is probably /opt/SUNWlu62, to LD_LIBRARY_PATH.

Basic configuration

1. Create the queue manager from the UNIX prompt using the command:
   ```
crtmqm -u dqname -q solaris
   ```
   where:
   - **solaris**
     Is the name of the queue manager
   - **-q**
     Indicates that this is to become the default queue manager
   - **-u dqname**
     Specifies the name of the undeliverable message queue

   This command creates a queue manager and a set of default objects.

2. Start the queue manager from the UNIX prompt using the command:
   ```
   strmqm solaris
   ```
   where **solaris** is the name given to the queue manager when it was created.

Channel configuration

The following section details the configuration to be performed on the Solaris queue manager to implement the channel described in Figure 32 on page 101.

The MQSC command to create each object is shown. Either start runmqsc from a UNIX prompt and enter each command in turn, or build the commands into a command file.

Examples are given for connecting WebSphere MQ for Solaris and MQSeries for OS/2 Warp. If you wish to connect to another WebSphere MQ product use the appropriate set of values from the table in place of those for OS/2.
Solaris configuration

Note: The words in **bold** are user-specified and reflect the names of WebSphere MQ objects used throughout these examples. If you change the names used here, ensure that you also change the other references made to these objects throughout this book. All others are keywords and should be entered as shown.

Table 30. Configuration worksheet for WebSphere MQ for Solaris

<table>
<thead>
<tr>
<th>ID</th>
<th>Parameter Name</th>
<th>Reference</th>
<th>Example Used</th>
<th>User Value</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><strong>Definition for local node</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>Queue Manager Name</td>
<td></td>
<td></td>
<td>SOLARIS</td>
</tr>
<tr>
<td>B</td>
<td>Local queue name</td>
<td></td>
<td></td>
<td>SOLARIS.LOCALQ</td>
</tr>
<tr>
<td></td>
<td><strong>Connection to MQSeries for OS/2 Warp</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>Remote queue manager name</td>
<td></td>
<td></td>
<td>OS2</td>
</tr>
<tr>
<td>D</td>
<td>Remote queue name</td>
<td></td>
<td></td>
<td>OS2.REMOTEQ</td>
</tr>
<tr>
<td>E</td>
<td>Queue name at remote system</td>
<td></td>
<td></td>
<td>OS2.LOCALQ</td>
</tr>
<tr>
<td>F</td>
<td>Transmission queue name</td>
<td></td>
<td></td>
<td>OS2</td>
</tr>
<tr>
<td>G</td>
<td>Sender (SNA) channel name</td>
<td></td>
<td></td>
<td>SOLARIS.OS2.SNA</td>
</tr>
<tr>
<td>H</td>
<td>Sender (TCP/IP) channel name</td>
<td></td>
<td></td>
<td>SOLARIS.OS2.TCP</td>
</tr>
<tr>
<td>I</td>
<td>Receiver (SNA) channel name</td>
<td></td>
<td></td>
<td>OS2.SOLARIS.SNA</td>
</tr>
<tr>
<td>J</td>
<td>Receiver (TCP/IP) channel name</td>
<td></td>
<td></td>
<td>OS2.SOLARIS.TCP</td>
</tr>
<tr>
<td></td>
<td><strong>Connection to WebSphere MQ for Windows</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>Remote queue manager name</td>
<td></td>
<td></td>
<td>WINNT</td>
</tr>
<tr>
<td>D</td>
<td>Remote queue name</td>
<td></td>
<td></td>
<td>WINNT.REMOTEQ</td>
</tr>
<tr>
<td>E</td>
<td>Queue name at remote system</td>
<td></td>
<td></td>
<td>WINNT.LOCALQ</td>
</tr>
<tr>
<td>F</td>
<td>Transmission queue name</td>
<td></td>
<td></td>
<td>WINNT</td>
</tr>
<tr>
<td>G</td>
<td>Sender (SNA) channel name</td>
<td></td>
<td></td>
<td>SOLARIS.WINNT.SNA</td>
</tr>
<tr>
<td>H</td>
<td>Sender (TCP/IP) channel name</td>
<td></td>
<td></td>
<td>SOLARIS.WINNT.TCP</td>
</tr>
<tr>
<td>I</td>
<td>Receiver (SNA) channel name</td>
<td></td>
<td></td>
<td>WINNT.SOLARIS.SNA</td>
</tr>
<tr>
<td>J</td>
<td>Receiver (TCP/IP) channel name</td>
<td></td>
<td></td>
<td>WINNT.SOLARIS.TCP</td>
</tr>
<tr>
<td></td>
<td><strong>Connection to WebSphere MQ for AIX</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>Remote queue manager name</td>
<td></td>
<td></td>
<td>AIX</td>
</tr>
<tr>
<td>D</td>
<td>Remote queue name</td>
<td></td>
<td></td>
<td>AIX.REMOTEQ</td>
</tr>
<tr>
<td>E</td>
<td>Queue name at remote system</td>
<td></td>
<td></td>
<td>AIX.LOCALQ</td>
</tr>
<tr>
<td>F</td>
<td>Transmission queue name</td>
<td></td>
<td></td>
<td>AIX</td>
</tr>
<tr>
<td>G</td>
<td>Sender (SNA) channel name</td>
<td></td>
<td></td>
<td>SOLARIS.AIX.SNA</td>
</tr>
<tr>
<td>H</td>
<td>Sender (TCP) channel name</td>
<td></td>
<td></td>
<td>SOLARIS.AIX.TCP</td>
</tr>
<tr>
<td>I</td>
<td>Receiver (SNA) channel name</td>
<td></td>
<td></td>
<td>AIX.SOLARIS.SNA</td>
</tr>
<tr>
<td>J</td>
<td>Receiver (TCP) channel name</td>
<td></td>
<td></td>
<td>AIX.SOLARIS.TCP</td>
</tr>
<tr>
<td></td>
<td><strong>Connection to MQSeries for Compaq Tru64 UNIX</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>Remote queue manager name</td>
<td></td>
<td></td>
<td>DECUX</td>
</tr>
<tr>
<td>D</td>
<td>Remote queue name</td>
<td></td>
<td></td>
<td>DECUX.REMOTEQ</td>
</tr>
<tr>
<td>E</td>
<td>Queue name at remote system</td>
<td></td>
<td></td>
<td>DECUX.LOCALQ</td>
</tr>
<tr>
<td>F</td>
<td>Transmission queue name</td>
<td></td>
<td></td>
<td>DECUX</td>
</tr>
</tbody>
</table>
### Table 30. Configuration worksheet for WebSphere MQ for Solaris (continued)

<table>
<thead>
<tr>
<th>ID</th>
<th>Parameter Name</th>
<th>Reference</th>
<th>Example Used</th>
<th>User Value</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Sender (TCP) channel name</td>
<td>DECUX.SOLARIS.TCP</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Receiver (TCP) channel name</td>
<td>SOLARIS.DECUX.TCP</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Connection to WebSphere MQ for HP-UX**

The values in this section of the table must match those used in Table 25 on page 239 as indicated.

| C   | Remote queue manager name | A                      | HPUX                        |                     |
| D   | Remote queue name         |                        | HPUX.REMOTEQ                |                     |
| E   | Queue name at remote system | B                    | HPUX.LOCALQ                 |                     |
| F   | Transmission queue name   |                        | HPUX                        |                     |
| G   | Sender (SNA) channel name |                         | SOLARIS.HPUX.SNA            |                     |
| H   | Sender (TCP) channel name |                         | SOLARIS.HPUX.TCP            |                     |
| I   | Receiver (SNA) channel name | C                   | HPUX.SOLARIS.SNA            |                     |
| J   | Receiver (TCP/IP) channel name | D              | HPUX.SOLARIS.TCP            |                     |

**Connection to MQSeries for AT&T GIS UNIX**

The values in this section of the table must match those used in Table 27 on page 253 as indicated.

| C   | Remote queue manager name | A                      | GIS                         |                     |
| D   | Remote queue name         |                        | GIS.REMOTEQ                 |                     |
| E   | Queue name at remote system | B                    | GIS.LOCALQ                  |                     |
| F   | Transmission queue name   |                        | GIS                         |                     |
| G   | Sender (SNA) channel name |                         | SOLARIS.GIS.SNA             |                     |
| H   | Sender (TCP/IP) channel name | H                   | SOLARIS.GIS.TCP             |                     |
| I   | Receiver (SNA) channel name | E                   | GIS.SOLARIS.SNA             |                     |
| J   | Receiver (TCP/IP) channel name | F               | GIS.SOLARIS.TCP             |                     |

**Connection to WebSphere MQ for iSeries**

The values in this section of the table must match those used in Table 49 on page 535 as indicated.

| C   | Remote queue manager name | A                      | AS400                       |                     |
| D   | Remote queue name         |                        | AS400.REMOTEQ               |                     |
| E   | Queue name at remote system | B                    | AS400.LOCALQ                |                     |
| F   | Transmission queue name   |                        | AS400                       |                     |
| G   | Sender (SNA) channel name |                         | SOLARIS.AS400.SNA           |                     |
| H   | Sender (TCP) channel name |                         | SOLARIS.AS400.TCP           |                     |
| I   | Receiver (SNA) channel name | G                   | AS400.SOLARIS.SNA           |                     |
| J   | Receiver (TCP) channel name | H                   | AS400.SOLARIS.TCP           |                     |

**Connection to WebSphere MQ for z/OS without CICS**

The values in this section of the table must match those used in Table 34 on page 372 as indicated.

| C   | Remote queue manager name | A                      | MVS                         |                     |
| D   | Remote queue name         |                        | MVS.REMOTEQ                 |                     |
| E   | Queue name at remote system | B                    | MVS.LOCALQ                  |                     |
| F   | Transmission queue name   |                        | MVS                         |                     |
| G   | Sender (SNA) channel name |                         | SOLARIS.MVS.SNA             |                     |
| H   | Sender (TCP) channel name |                         | SOLARIS.MVS.TCP             |                     |
| I   | Receiver (SNA) channel name | G                   | MVS.SOLARIS.SNA             |                     |
| J   | Receiver (TCP) channel name | H                   | MVS.SOLARIS.TCP             |                     |
Solaris configuration

Table 30. Configuration worksheet for WebSphere MQ for Solaris (continued)

<table>
<thead>
<tr>
<th>ID</th>
<th>Parameter Name</th>
<th>Reference</th>
<th>Example Used</th>
<th>User Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>C</td>
<td>Remote queue manager name</td>
<td></td>
<td>VSE</td>
<td></td>
</tr>
<tr>
<td>D</td>
<td>Remote queue name</td>
<td></td>
<td>VSE.REMOTEQ</td>
<td></td>
</tr>
<tr>
<td>E</td>
<td>Queue name at remote system</td>
<td></td>
<td>VSE.LOCALQ</td>
<td></td>
</tr>
<tr>
<td>F</td>
<td>Transmission queue name</td>
<td></td>
<td>VSE</td>
<td></td>
</tr>
<tr>
<td>G</td>
<td>Sender channel name</td>
<td></td>
<td>SOLARIS.VSE.SNA</td>
<td></td>
</tr>
<tr>
<td>H</td>
<td>Receiver channel name</td>
<td></td>
<td>VSE.SOLARIS.SNA</td>
<td></td>
</tr>
</tbody>
</table>

WebSphere MQ for Solaris sender-channel definitions using SNA using SunLink Version 9.1

```bash
def q1 (OS2) +     
   usage(xmitq) + 
   replace

def qr (OS2.REMOTEQ) +    
   rname(OS2.LOCALQ) + 
   rqmname(OS2) + 
   xmitq(OS2) + 
   replace

def chl (SOLARIS.OS2.SNA) chltype(sdr) +    
   trptype(lu62) + 
   connname('/home/mqstart/OS2CPIC') + 
   xmitq(OS2) + 
   replace
```

WebSphere MQ for Solaris sender-channel definitions using SNAP-IX SNA

```bash
def q1 (OS2) +     
   usage(xmitq) + 
   replace

def qr (OS2.REMOTEQ) +    
   rname(OS2.LOCALQ) + 
   rqmname(OS2) + 
   xmitq(OS2) + 
   replace

def chl (SOLARIS.OS2.SNA) chltype(sdr) +    
   trptype(lu62) + 
   connname('OS2CPIC') + 
   xmitq(OS2) + 
   replace
```

WebSphere MQ for Solaris receiver-channel definitions using SNA

```bash
def q1 (SOLARIS.LOCALQ) replace

def chl (OS2.SOLARIS.SNA) chltype(rcvr) +    
   trptype(lu62) + 
   replace
```
WebSphere MQ for Solaris sender-channel definitions using TCP

def ql (OS2) +
  usage(xmitq) +
  replace

def qr (OS2.REMOTEQ) +
  rname(OS2.LOCALQ) +
  rqnname(OS2) +
  xmitq(OS2) +
  replace

def chl (SOLARIS.OS2.TCP) chltype(sdr) +
  trptype(tcp) +
  conname(remote_tcpp_ip_hostname) +
  xmitq(OS2) +
  replace

WebSphere MQ for Solaris receiver-channel definitions using TCP/IP

def ql (SOLARIS.LOCALQ) replace

def chl (OS2.SOLARIS.TCP) chltype(rcvr) +
  trptype(tcp) +
  replace
Solaris configuration
Chapter 19. Setting up communication in Compaq OpenVMS Alpha systems

Distributed queue management (DQM) is a remote queuing facility for WebSphere MQ. It provides channel control programs for the queue manager that form the interface to communication links, controllable by the system operator. The channel definitions held by distributed-queuing management use these connections.

When a distributed-queuing management channel is started, it tries to use the connection specified in the channel definition. For this to succeed, it is necessary for the connection to be defined and available. This chapter explains how to do this.

For OS/2 and Windows NT, see Chapter 10, “Setting up communication for OS/2 and Windows” on page 129. For UNIX systems, see Chapter 13, “Setting up communication in UNIX systems” on page 191. For Compaq NonStop Kernel, see Chapter 20, “Setting up communication in Compaq NonStop Kernel” on page 307.

Deciding on a connection

There are four forms of communication for WebSphere MQ on Compaq OpenVMS Alpha systems:

• TCP
• LU 6.2
• DECnet Phase IV
• DECnet Phase V

Each channel definition must specify one only as the transmission protocol (Transport Type) attribute. One or more protocols may be used by a queue manager.

For WebSphere MQ clients, it may be useful to have alternative channels using different transmission protocols. See the WebSphere MQ Clients book.
Defining a TCP connection

The channel definition at the sending end specifies the address of the target. The TCP service is configured for the connection at the receiving end.

**Sending end**

Specify the host name, or the TCP address of the target machine, in the Connection Name field of the channel definition. Port number 1414 is assigned by the Internet Assigned Numbers Authority to WebSphere MQ.

To use a port number other than the default, change the connection name field thus:

```
Connection Name REMHOST(1822)
```

where REMHOST is the hostname of the remote machine and 1822 is the port number required. (This must be the port that the listener at the receiving end is listening on.)

Alternatively you can change the default sending port number by specifying it in the queue manager configuration file (qm.ini):

```
TCP:
  Port=1822
```

For more information about the values you set using qm.ini, see Appendix D, “Configuration file stanzas for distributed queuing” on page 715.

**Receiving channels using Compaq (DIGITAL) TCP/IP services (UCX) for OpenVMS**

To use Compaq (DIGITAL) TCP/IP Services (UCX) for OpenVMS, you must configure a UCX service as follows:

1. Create a file consisting of one line and containing the DCL command to start the TCP/IP receiver program, amqcrsta.exe:

   ```
   $ mcr amqcrsta [-m Queue_Man_Name]
   ```

   Place this file in the SYS$MANAGER directory. In this example the name of the file is MQRECV.COM.

**Notes:**

a. If you have multiple queue managers you must make a new file and UCX service for each queue manager.

b. Ensure that the protection on the file and its parent directory allow it to be executable, that is, the protection is /PROT=W:RE.

2. Create a UCX service to start the receiving channel program automatically:

   ```
   $ UCX
   UCX> set service <p1>/port=<p2>/protocol=TCP/user_name=MQM -
   ```

   where:

   **p1** Is the service name, for example MQSERIES01. A unique name is required for each queue manager defined.

   **p2** Is the TCP/IP port number in the range 1 to 65 535. The default value for WebSphere MQ is 1414.
Defining a TCP connection

p3 Is the process name. This consists of a string up to 15 characters long.
p4 Is the name of the startup command file, for example, SYS$MANAGER:MQRECV.COM.
p5 Is the process limit. This is the maximum number of connections allowed using the port number. If this limit is reached, subsequent requests are rejected.

Note: Each channel represents a single connection to the queue manager.

If a receiving channel does not start when the sending end starts, it is probably due to the permissions on the file being incorrect.

3. To enable the service upon every system IPL (reboot), issue the command

$ UCX SET CONFIGURATION ENABLE SERVICE MQSERIES

Using the TCP/IP SO_KEEPALIVE option
If you want to use the SO_KEEPALIVE option (as discussed in “Checking that the other end of the channel is still available” on page 66) you must add the following entry to your queue manager configuration file (qm.ini) or the Windows NT registry:

TCP:
KeepAlive=yes

Receiving channels using Cisco MultiNet for OpenVMS
To use Cisco MultiNet for OpenVMS, you must configure a MultiNet service as follows:

1. Create a file consisting of one line and containing the DCL command to start the TCP receiver program, amqcrsta.exe:
   
   $ mcr amqcrsta.exe [-m Queue_Man_Name]

   Place this file in the SYS$MANAGER directory.

   Notes:
   a. If you have multiple queue managers you must make a new file and MultiNet service for each queue manager.
   b. Ensure that the protection on the file and its parent directory allow it to be executable, that is, the protection is /PROT=W:RE.

2. Create a MultiNet service to start the receiving channel program automatically:

   $ multinet configure/server
   MultiNet Server Configuration Utility 3.5 (101)
   [Reading in configuration from MULTINET:SERVICES.MASTER_SERVER]
   SERVER-CONFIG> add WebSphere MQ
   [Adding new configuration entry for service "MQSERIES"]
   Protocol: [TCP]
   TCP Port number: 1414
   Program to run: sys$manager:mqrecv.com
   [Added service MQSERIES to configuration]
   [Selected service is now MQSERIES]
   SERVER-CONFIG> set flags UCX_SERVER
   MQSERIES flags set to <UCX_SERVER>
   SERVER-CONFIG> set username MQM
   [Username for service MQSERIES set to MQM]
   SERVER-CONFIG> exit
   [Writing configuration to MULTINET_COMMON_ROOT:SERVICES.MASTER_SERVER]
Defining a TCP connection

The service is enabled automatically after the next system IPL (reboot). To enable the service immediately, issue the command

'MULTINET
CONFIGURE /SERVER RESTART'.

Receiving channels using Attachmate PathWay for OpenVMS

To use Attachmate PathWay for OpenVMS to start channels, you must configure a PathWay service as follows:

1. Create a file consisting of one line and containing the DCL command to start the TCP/IP receiver program, amqcrsta.exe:

   $ mcr amqcrsta [-m Queue_Manager_Name]

   Place this file in the SYS$MANAGER directory. In this example the name mqrecv.com is used.

2. Create an Attachmate service to start the receiving channel program automatically.

   You do this by adding the following lines to the file

   TWGS$COMMON:[NETDIST.ETC]SERVERS.DAT.

   # MQSeries
   service-name MQSeries
   program SYS$MANAGER:MQRECV.COM
   socket-type SOCK_STREAM
   socket-options SO_ACCEPTCONN | SO_KEEPALIVE
   socket-address AF_INET , 1414
   working-set 512
   priority 4
   INIT TCP_Init
   LISTEN TCP_Listen
   CONNECTED TCP_Connected
   SERVICE Run_Program
   username MQM
   device-type UCX

Receiving channels using Process Software Corporation TCPware

To use Process Software Corporation TCPware, you must configure a TCPware service as follows:

1. Create a file consisting of one line and containing the DCL command to start the TCP receiver program amqcrsta.exe:

   $ mcr amqcrsta (-m Queue_Manager_Name)

   Place this file in the SYS$MANAGER directory. In this example the name of the file is MQRECV.COM.

   Notes:
   a. If you have multiple queue managers you must make a new file and TCPware service for each queue manager.
   b. Ensure that the protection on the file and its parent directory allow it to be executable, that is, the protection is /PROT=W:RE.

2. Create a TCPware service to start the receiving channel program automatically:

   a. Edit the TCPWARE:SERVICES. file and add an entry for the service you want to use:

      MQSeries 1414/tcp # MQSeries port
b. Edit the TCPWARE:SERVERS.COM file and add an entry for the service defined in the previous step:

```
$! SERVERS.COM
$!
$ RUN TCPWARE:NETCU
ADD SERVICE MQSeries BG_TCP -
     /INPUT=SYS$MANAGER:MQRECV.COM -
     /LIMIT=6 -
     /OPTION=KEEPALIVE -
     /USERNAME=MQM
EXIT
```

3. The service is enabled automatically after the next system IPL. To enable the service immediately issue the command:

```
@TCPWARE:SERVERS.COM
```

---

### Defining an LU 6.2 connection

MQSeries for Compaq OpenVMS Alpha uses the DECnet SNA APPC/LU 6.2 Programming Interface. This interface requires access through DECnet to a suitably configured SNA Gateway, for example, the SNA Gateway-ST, or SNA Gateway-CT.

#### SNA configuration

To enable WebSphere MQ to work with DECnet APPC/LU 6.2 you must complete your Gateway SNA configuration first. The Digital SNA configuration must be in agreement with the Host SNA configuration.

**Notes:**

1. When configuring your host system, be aware that the DECnet SNA Gateway supports PU 2.0 and not node type 2.1. This means that the LUs on the Digital SNA node must be dependent LUs. They reside on the Digital SNA node and so must be defined and configured there. However, because they are dependent LUs, they have to be activated by VTAM, by means of an ACTLU command, and so they also need to be defined to VTAM as dependent LUs.

2. Ensure that the SNA libraries are installed as shared images upon each system IPL by running the command @SYS$STARTUP:SNALU62$STARTUP.COM in the system startup procedure.

To configure your SNA Gateway, set up the SNAGATEWAY_<node>_SNA.COM file, where <node> is replaced with the node name of your DECnet SNA gateway.

Do this by responding to the configuration prompts in the Gateway installation procedure, or by directly editing the file.

The SNA Gateway installation procedure creates the file in the directory SYSS$COMMON:[SNA$CSV].

The configuration information in this file is downloaded to the Gateway when you run the NCP LOAD NODE command.

**Notes:**

1. Online changes to the current Gateway configuration can be made using the utility SNANCP.

2. SNA resources can be monitored using the SNAP utility.
A sample SNA Gateway Configuration file follows:

```
$!+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
$! Start of file: SYS$COMMON:[SNA$CSV]SNAGATEWAY_SNAGWY_SNA.COM
$! DECnet SNA Gateway-ST SNA configuration file
$! Created: 23-FEB-1996 19:10:43.68 by SNACST$CONFIGURE V1.2
$! Host node: CREAMP User$ CHO
$!+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
$ v = $verify(1)
$ RUN SYS$SYSTEM:SNANCP
SET LINE SYN-0 - // Line definition
   DUPLEX FULL -
   PROTOCOL SDLC POINT -
   SIGNALLING NORMAL -
   CLOCK EXTERNAL -
   MODEM TYPE NORMAL -
   RECEIVE BUFFERS 34 -
   LOGGING INFORMATIONAL -
   BUFFER SIZE 265
SET CIRCUIT SDLC-0 - // Circuit definition
   LINE SYN-0 -
   DUPLEX FULL -
   RESPONSE MODE NORMAL -
   STATION ADDRESS C1 -
   LOGGING INFORMATIONAL -
   STATION ID 0714002A // XID
SET PU SNA-0 CIRCUIT SDLC-0 -
   LU LIST 1-32 -
   SEGMENT SIZE 265 - // must equal MAXDATA on Host PU definition
   LOGGING WARNING
SET CIRCUIT SDLC-0 STATE ON
SET LINE SYN-0 STATE ON
SET SERVER SNA-ACCESS -
   LOGGING WARNING -
   NOTE "Gateway Access Server" -
   STATE ON
SET ACCESS NAME VTAMSDR PU SNA-0 LU 2 APPL MVSLU LOGON LU62SS
SET ACCESS NAME VTAMRCVR PU SNA-0 LU 3 APPL MVSLU LOGON LU62SS
$ EXIT $STATUS + (0 * $verify(v))
$!+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
$! End of file: SYS$COMMON:[SNA$CSV]SNAGATEWAY_SNAGWY_SNA.COM
$!+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

**Defining access names**

You should set up a separate Access name for each WebSphere MQ channel. This ensures that the VMS system and the remote system agree on the LU used for the channel.

**Note:** If you use a single access name, with a range of LUs specified, the Gateway selects the LUs in a circular order. Therefore the LU selected by the Gateway may not correspond with the LU used by the Host channel, because the Host associates a specific LU with a channel.

The access name is used only to communicate between the DECnet SNA APPC program and the Gateway. It has no network meaning.

**Notes:**

1. The LUs are single session. You **must** define a separate LU for each channel if they are to run simultaneously.
2. You are advised to use names that associate the access name to the corresponding channel, but you can choose any name.
3. The APPL in the ACCESS name definition must match the remote (in this case MVSLU) APPL in VTAM.
4. The LU number must correspond to the LOCADDR in the LU definition statement in VTAM. Here is an example VTAM line and LU definitions:
5. The LOGON must specify the logmode entry on the VTAM host that specifies parameters acceptable to the SNA Gateway. Here is an example of a single session logmode entry:

```
LU62SS MODEENT LOGMODE=LU62SS,
    TYPE=0,        ONLY TYPE RECOGNIZED
    FMPROF=X'13',  SNA
    TSPROF=X'07',  SNA
    PRIPROT=X'80', PRIMARY PROTOCOL
    SECPROT=X'80', SECONDARY PROTOCOL
    COMPROT=X'5081', COMMON PROTOCOL
    SSNDPAC=X'00',
    SRCVPAC=X'00',
    RUSIZES=X'8989', RUSIZES IN-4096 OUT-4096
    PSNDPAC=X'00',
    PSERVIC=X'000000000000000000002C00'.
```

The *DECnet SNA Gateway Guide to IBM Parameters* details the parameters expected by the Gateway.

### Specifying SNA configuration parameters to WebSphere MQ

WebSphere MQ obtains knowledge of the SNA resources by passing the Gateway Node name and the Access name to the channel program.

#### Passing parameters to sender and requester channel pairs

For sender and requester channel pairs specify the Gateway Node and Access Name in the CONNAME string in the channel definition.

The CONNAME also includes the TPNAME that is used by the SNA Allocate verb to invoke the remote program.

The format of the CONNAME is: `CONNAME('GatewayNode.AccessName(TpName)').`

For example: `CONNAME('SNAGWY.VTAMSDR(MQSERIES)'),` where SNAGWY is the Gateway node, VTAMSDR is the access name, and WebSphere MQ is the TPNAME.

**Note:** Do not use the TPNAME field in the channel definition.

### Running senders and requesters

Senders, requesters, and fully qualified servers can be explicitly run by performing a START CHANNEL command in runmqsc.

Senders and requesters on Compaq OpenVMS Alpha initiate a session by issuing an INIT-SELF to request a BIND from the host. In issuing the Allocate verb, the WebSphere MQ channel program takes the LU name and the Mode Name from the Access Name.

WebSphere MQ then allocates a conversation using the specified TPNAME.
**Defining an LU 6.2 connection**

**Passing parameters to servers and receivers**
For servers and receivers, specify the Gateway Node, Access Name, and TPNAME as command line parameters to the `runmqlsr` command.

**Running servers and receivers**
Servers and receivers are started by running the `runmqlsr` command.

```bash
$ RUNMQLSR -m QMName -n TPname -g GatewayNode(AccessName)
```

**Note:** Each server and receiver channel requires its own listener process.

You can include these commands in the WebSphere MQ startup file, `SYS$STARTUP:MQS_STARTUP`.

Receivers and servers issue the ACTIVATE_SESSION request to the Gateway in passive mode. In passive mode the channel program waits for a BIND from the remote system, which puts the LU into the active-listening state, waiting for a bind from the host.

You can check the LU status using SNANCP to make sure that you are in active-listening state on the correct LU. If a BIND from the host arrives specifying the LU that is in active-listening state, the session will be established. After establishing the session, the host attempts to allocate a conversation.

The TPNAME used by the host sender/requester channel *must* be the same name as that specified on the command line in order to establish the conversation.

**Note:** `RUNMQLSR` recycles when a remote channel disconnects. There is a finite period of time before the listener is ready to accept further binds from the host.

**Ending the SNA Listener process**
To find the batch job number for the SNA listener process, type: `$ show queue /all`

To end the SNA Listener process type:

```bash
$ delete /entry=<jobnumber>
```

where `<jobnumber>` is the job number of the listener batch job.

**Sample WebSphere MQ configuration**

```bash
* channel configuration for saturn.queue.manager for LU6.2
* def q1('HOST_SENDER_TQ') usage(xmitq)
def chl('HOST.TO.VMS') chltype(rcvr) trptype(lu62) + seqwrap(999999999)
def chl('VMS.TO.HOST') chltype(sdr) trptype(lu62) + conname('SNAGWY.VTAMSDR(MQSERIES)') + xmitq('HOST_SENDER_TQ') seqwrap(999999999)
```

In this example two channels, a sender and a receiver, have been set up.

On the remote system you need to configure the corresponding channels. Channels that talk to each other must have the same name.
Defining an LU 6.2 connection

- The OpenVMS sender, VMS.TO.HOST, talks to a receiver called VMS.TO.HOST on the host system.
- The OpenVMS receiver, HOST.TO.VMS talks to a sender HOST.TO.VMS on the host system.

The commands to start each channel are:

// Start sender channel to host system
$ runmqchl -m "saturn.queue.manager" -c "VMS.TO.HOST"
// Set up listener to listen for incoming SNA requests.
$ runmqlsr -m "saturn.queue.manager" -n "TPNAME" -g SNAGWY(VTAMRCVR)

**Note:** The TPNAME must match the outbound TPNAME on the z/OS sender channel side. This is specified in the z/OS side information, for example:

```
SIDELETE
DESTNAME(ID1)
STADD
DESTNAME(ID1)
MODENAME(LU62SS)
TPNAME(MQSERIES)
PARTNER_LU(IYA83072)
```

**Problem solving**

**Error PUNOTAVA - PU has not been activated**

This error indicates a lack of connectivity between the two machines. Make sure your line and circuit are set to state ON. Use SNATRACE at the circuit level to verify that the Compaq OpenVMS Alpha machine is polling. If no response is received for the poll, check that the PU on the host is enabled. If the line will not go to the ON STATE check your physical line. If the trace shows the host responding to the poll, but the PU still does not become active, check your setting of the STATION ID.

**Failure to allocate conversation**

This error is returned by a sender or requester to indicate that allocate failed. Run trace to verify that the session can be established. Verify that the Compaq OpenVMS Alpha machine sends the INIT-SELF (010681). If there is no response to the INIT-SELF make sure that the host WebSphere MQ channel is started. If the BIND from the host is rejected by the Compaq OpenVMS Alpha machine analyze the Digital bind response. Use the *DECnet SNA Gateway Guide to IBM Parameters* to see what is set incorrectly in the mode. If a session is established and the conversation allocate request is rejected verify that the TPNAMEs are configured the same on both systems.

For receivers and servers verify that a BIND is sent by the host. If not, enable the Host WebSphere MQ channel. If the BIND is rejected check the reason for rejection. Make sure that the Compaq OpenVMS Alpha listener LU is the LU with which the host is trying to establish a session.

**WebSphere MQ connection failure**

After establishing a conversation the two WebSphere MQ channels engage in a protocol to establish a WebSphere MQ channel connection. If this fails, the reason for failure should be indicated in the error logs on the two systems. Check both logs and correct the indicated problem. For example the connection fails if one system has a SEQWRAP value of 999999999 and the other 999999. In the
Defining an LU 6.2 connection

SNATRACE you will see that the allocate succeeded and that MQ is trying to establish a channel connection. At this point the WebSphere MQ logs are the best aid in resolving problems.

Defining a DECnet Phase IV connection

The channel definition at the sending end specifies the address of the target. The DECnet network object is configured for the connection at the receiving end.

Sending end

Specify the DECnet node name and the DECNET object name in the Connection Name field of the channel definition. You need a different DECnet object for each separate queue manager that is defined. For example, to specify DECnet object MQSERIES on node FOONT enter the following when defining the channel:

CONNAME('FOONT(MQSERIES)')

Receiving on DECnet Phase IV

To use DECnet Phase IV to start channels, you must configure a DECnet object as follows:

1. Create a file consisting of one line and containing the DCL command to start the DECnet receiver program, amqcrsta.exe:

   $ mcr amqcrsta [-m Queue_Man_Name] -t DECnet

   Place this file in the SYS$MANAGER directory. In this example the file is named MQRECVDECNET.COM.

   Notes:
   a. If you have multiple queue managers you must make a new file and DECnet object for each queue manager.
   b. If a receiving channel does not start when the sending end starts, it is probably due to the permissions on this file being incorrect.

2. Create a DECnet object to start the receiving channel program automatically. You must supply the correct password for WebSphere MQ.

   $ MCR NCP
   NCP> define object MQSERIES
   Object number (0-255): 0
   File name (filename):sys$manager:mqrecvdecnet.com
   Privileges (List of VMS privileges):
   Outgoing connect privileges (List of VMS privileges):
   User ID (1-39 characters): mqm
   Password (1-39 characters): mqseries
   Account (1-39 characters):
   Proxy access (INCOMING, OUTGOING, BOTH, NONE, REQUIRED): ALL
   NCP> set known objects all
   NCP> exit

   Note: You could use proxy user identifiers rather than actual user identifiers. This will prevent any unauthorized access to the database. Information on how to set up proxy identifiers is given in the Digital DECnet for OpenVMS Networking Manual.

3. Ensure that all known objects are set when DECnet is started.

Defining a DECnet Phase V connection

Set up the WebSphere MQ configuration for channel objects:

1. Start the NCL configuration interface by issuing the following command:
2. Create a session control application entity by issuing the following commands:

```ncl
NCL> create session control application MQSERIES
NCL> set sess con app MQSERIES address {name=MQSERIES}
NCL> set sess con app MQSERIES image name -
   _SYS$MANAGER:MQRECVDECNET.COM
NCL> set sess con app MQSERIES user name "MQM"
NCL> set sess con app MQSERIES node synonym true
NCL> show sess con app MQSERIES all [characteristics]
```

**Note:** User-defined values are in **uppercase**.

3. Create the command file as for DECnet PhaseIV.

4. The log file for the object is net$server.log in the sys$login directory for the application-specified user name.

5. To enable the session control application upon every system IPL (reboot), add the preceding NCL commands to the file `SYS$MANAGER:NET$APPLICATION_LOCAL.NCL`. 

DECnt Phase V connections
DECnet Phase V connections
Distributed queue management (DQM) is a remote queuing facility for WebSphere MQ. It provides channel control programs for the queue manager that form the interface to communication links, controllable by the system operator. The channel definitions held by distributed-queuing management use these connections.

When a distributed-queuing management channel is started, it tries to use the connection specified in the channel definition. For this to succeed, it is necessary for the connection to be defined and available. This chapter explains how to do this.

For OS/2 and Windows NT, see Chapter 10, “Setting up communication for OS/2 and Windows” on page 129. For UNIX systems, see Chapter 13, “Setting up communication in UNIX systems” on page 191. For Compaq OpenVMS Alpha, see Chapter 19, “Setting up communication in Compaq OpenVMS Alpha systems” on page 295.

Deciding on a connection

There are two forms of communication for MQSeries for Compaq NonStop Kernel:

- TCP
- LU 6.2

Each channel definition must specify one only as the transmission protocol (Transport Type) attribute. One or more protocols may be used by a queue manager.

When connecting to WebSphere MQ clients, it may be useful to have alternative channels using different transmission protocols. See the WebSphere MQ Clients book for more information. (There is no MQSeries for Compaq NonStop Kernel client.)

SNA channels

The following channel attributes are necessary for SNA channels in MQSeries for Compaq NonStop Kernel:

**CONNAME**

The value of CONNAME depends on whether SNAX or ICE is used as the communications protocol:

*If SNAX is used:*

CONNAME(‘$PPPP.LOCALLU.REMOTELU’)

Applies to sender, requester, and fully-qualified server channels, where:

$PPPP  Is the process name of the SNAX/APC process.
LOCALLU  Is the name of the Local LU.
REMOTELU  Is the name of the partner LU on the remote machine.

For example:
SNA channels

**CONNAME**('$BP01.IYAHT080.IYCNVM03')

**CONNAME**('$$PPP.LOCALLU')

Applies to receiver and non fully-qualified server channels, where:

$PPP  Is the process name of the SNAX/APC process.

LOCALU  Is the name of the Local LU. This value can be an asterisk (*), indicating any name.

For example:

CONNAME('$$BP01.IYAHT080')

*If ICE is used:*

**CONNAME**('$$PPP.#OPEN.LOCALLU.REMOTELU')

Applies to sender, requester, and fully-qualified server channels, where:

$PPP  Is the process name of the ICE process.

#OPEN  Is the ICE open name.

LOCALU  Is the name of the Local LU.

REMOTELU  Is the name of the partner LU on the remote machine.

For example:

CONNAME('$$ICE.#IYAHT0C.IYAHT0C0.IYCNVM03')

**MODENAME**

Is the SNA mode name. For example, MODENAME(LU62PS).

**TPNAME**('LOCALTP[.REMOTETP]')

Is the Transaction Process (TP) name.

LOCALTP  Is the local name of the TP.

REMOTETP  Is the name of the TP on the remote machine. This value is optional. If it is not specified, and the channel is one that initiates a conversation (that is, a sender, requester, or fully-qualified server channel) the LOCALTP name is used.

Both the LOCALTP and REMOTETP values can be up to 16 characters in length.
Notes:

1. If SNAX is being used to facilitate SNA communications, the values in the LOCALTP field in the TPNAME must match TPs defined to SNAX. You are recommended to use uppercase when defining an LU name.

2. If ICE is being used, TPNAMEs do not need to be defined to ICE; they need only be present in the WebSphere MQ channel definitions.

LU 6.2 responder processes

There is no SNA listener process in MQSeries for Compaq NonStop Kernel. Each channel initiated from a remote system (receiver, server, or requester that has a fully-qualified server on the remote system or a requester that has a sender on the remote system) must have its own, unique TP name on which it can listen. This TP name is specified as the LOCALTP value.

Such channels must be defined to MQSC with the attribute AUTOSTART(ENABLED) to ensure that there is an LU 6.2 responder process listening on this TP name whenever the queue manager is started. This LU 6.2 responder process (MQLU6RES) services incoming SNA requests for its particular TP. If the channel is newly defined, or has been recently altered, an LU 6.2 responder process can be started for that channel by issuing either the MQSC command START CHANNEL (using runmqsc) or the runmqch1 control command from the TACL prompt.

SNA channels defined AUTOSTART(DISABLED) do not listen for incoming SNA requests. LU 6.2 responder processes are not started for such channels. A message is logged to MQERRLG1 whenever an LU 6.2 responder process is started.

TCP channels

For information about using a nondefault TCP process for communications via TCP, and information about the TCP ports a queue manager listens on, see the MQSeries for Compaq NonStop Kernel System Administration.

Communications examples

This section provides communications setup examples for SNA (SNAX and ICE) and TCP.

SNAX communications example

This section provides:

- An example SCF configuration file for the SNA line
- Some example SYSGEN parameters to support the line
- An example SCF configuration file for the SNA process definition
- Some example MQSC channel definitions

SCF SNA line configuration file

Here is an example SCF configuration file:
Communications examples

==
== SCF configuration file for defining SNA LINE, PUs, and LUs to VTAM
== Line is called $SNA02 and SYSGEN'd into the Tandem system
==

ALLOW ALL
ASSUME LINE $SNA02

ABORT, SUB LU
ABORT, SUB PU
ABORT

DELETE, SUB LU
DELETE, SUB PU
DELETE
ADD $SNA02 LINE DEFINITION

ADD LINE $SNA02, STATION SECONDARY, MAXPUS 5, MAXLUS 1024, RECSIZE 2048, &
CHARACTERSET ASCII, MAXLOCALUS 256, &
PUIDBLK %H05D, PUIDNUM %H312FB

ADD REMOTE PU OBJECT, LOCAL IS IMPLICITLY DEFINED AS #ZNT21

ADD PU #PU2, ADDRESS 1, MAXLUS 16, RECSIZE 2046, TYPE (13,21), &
TRRMTADDR 04400045121088, DYNAMIC ON, &
ASSOCIATESUBDEV $CHAMB.#p2, &
TRSSAP %H04, &
CPNAME IYAQCDRM, SNANETID GBIBMIYA

ADD LOCAL LU OBJECT

ADD LU #ZNTLU1, TYPE (14,21), RECSIZE 1024, &
CHARACTERSET ASCII, PUNAME #ZNT21, SNANAME IYAHT080

ADD PARTNER LU OBJECTS

spinach (HP)
ADD LU #PU2LU1, TYPE(14,21), PUNAME #PU2, SNANAME IYABTOF0

stingray (AIX)
ADD LU #PU2LU2, TYPE(14,21), PUNAME #PU2, SNANAME IYA3T995

coop007 (OS/2)
ADD LU #PU2LU3, TYPE(14,21), PUNAME #PU2, SNANAME IYAFT170

MVS CICS
ADD LU #PU2LU4, TYPE(14,21), PUNAME #PU2, SNANAME IYCMVM03

MVS Non-CICS
ADD LU #PU2LU5, TYPE(14,21), PUNAME #PU2, SNANAME IYCNVM03

finnr100 (NT)
ADD LU #PU2LU6, TYPE(14,21), PUNAME #PU2, SNANAME IYAFTO80

winas18 (AS400)
ADD LU #PU2LU7, TYPE(14,21), PUNAME #PU2, SNANAME IYAF110

MQ-Portuguese (OS/2)
ADD LU #PU2LU8, TYPE(14,21), PUNAME #PU2, SNANAME IYAH090

VSE
ADD LU #PU2LU10, TYPE(14,21), PUNAME #PU2, SNANAME IYZMZSI2
SYSGEN parameters
The following are CONFTEXT file entries for a SYSGEN to support the SNA and
token ring lines:

== START UP TOKEN RING ASSOCIATE SUB DEVICE $CHAMB.#P2
== then start the line, pu's, and lu's
START LINE $CHAMB, SUB ALL
START
START, SUB PU
STATUS
STATUS, SUB PU
STATUS, SUB LU

SYSGEN parameters
The following are CONFTEXT file entries for a SYSGEN to support the SNA and
token ring lines:

!**************************************************************************
! LAN MACRO
!**************************************************************************
! This macro is used for all 361x LAN controllers
! REQUIRES T9375 SOFTWARE PACKAGE
C3613^MLAM = MLAM
    TYPE 56,
    SUBTYPE 0,
    PROGRAM C9376P00,
    INTERRUPT IOP^interrupt^handler,
    MAXREQUESTSIZE 32000,
    RSIZE 32000,
    BURSTSIZE 16,
    LINEBUFFERSIZE 32,
    STARTDOWN #;

!**************************************************************************
! SNAX macro for Token ring lines
!**************************************************************************
TOKEN^RING^SNAX^MACRO = SNATS
    TYPE 58,
    SUBTYPE 4,
    RSIZE 1024,
    SUBTYPE 4,
    FRAMESIZE 1036 # ;

!**************************************************************************
! SNAX MANAGER
!**************************************************************************
SSCP^MACRO = SNASVM
    TYPE 13,
    SUBTYPE 5,
    RSIZE 256 #;
Communications examples

SNAX/APC process configuration

The following definitions configure the example APC process (process name $BP01) via SCF for the SNA line.

Note: The pathway process $BP01 is created using the Tandem utility APCRUN.

SNAX/Token Ring Pseudocontroller
RING 6997 0,1 %360 ;

Token Ring Line
$CHAMB LAN1.0, LAN1.1 C3613^MLAM, NAME #LAN1;

Configure the SSCP
$SSCP SNAX.0, SNAX.1 SSCP^MACRO;

Sna lines for Dummy Controller over Token Ring
$SNA01 RING.0, RING.1 TOKEN^RING^SNAX^MACRO;
$SNA02 RING.2, RING.3 TOKEN^RING^SNAX^MACRO;

==
== SCF Configuration file for SNAX/APC Lus
==

ALLOW ERRORS

ASSUME PROCESS $BP01

ABORT SESSION *
ABORT TPN *
ABORT PTNR-MODE *
ABORT PTNR-LU *
ABORT LU *

DELETE TPN *
DELETE PTNR-MODE *
DELETE PTNR-LU *
DELETE LU *

==

ADD LOCAL LU

ADD LU IYAHT080, SNANAME GBIBMIYA.IYAHT080, SNAXFILENAME $SNA02.#ZNTLU1, &
MAXSESSION 256, AUTOSTART YES

== TPNames for MQSeries

ADD TPN IYAHT080.INTCRS6A
ADD TPN IYAHT080.DUMMY, GENERALTPREADY yes, SESSIONCONTROL yes, &
REMOTEATTACHTIMER -1, REMOTEATTACH queue
Communications examples

=== Spinach (HP) Partner LU
ADD PTNR-LU IYAHTO80.IYABTOF0, SNANAME GBIBMIYA.IYABTOF0, & PERIPHERAL-NODE NO, PARALLEL-SESSION-LU YES
ADD PTNR-MODE IYAHTO80.IYABTOF0.LU62PS, MODENAME LU62PS, & DEFAULTMAXSESSION 8, DEFAULTMINCONWINNER 4, & DEFAULTMINCONLOSER 3, MAXAUTOACT 1, RCVWINDOW 4, & DEFAULTMAXINRUSIZE 1024, DEFAULTMAXOUTRUSIZE 1024, & SENDWINDOW 4
ADD TPN IYAHTO80.MH01SDRCSDR
ADD TPN IYAHTO80.MH01RQSDSDR
ADD TPN IYAHTO80.MH01RQSVSVR
ADD TPN IYAHTO80.MH01SDRCCVR
ADD TPN IYAHTO80.MH01RQSVRQSTR
ADD TPN IYAHTO80.MH01RQSDRQSTR

==
== Winas18 (AS400) Partner LU
==
ADD PTNR-LU IYAHTO80.IYAFT110, SNANAME GBIBMIYA.IYAFT110, & PERIPHERAL-NODE NO, PARALLEL-SESSION-LU YES
ADD PTNR-MODE IYAHTO80.IYAFT110.LU62PS, MODENAME LU62PS, & DEFAULTMAXSESSION 8, DEFAULTMINCONWINNER 4, & DEFAULTMINCONLOSER 3, MAXAUTOACT 1, RCVWINDOW 4, & DEFAULTMAXINRUSIZE 1024, DEFAULTMAXOUTRUSIZE 1024, & SENDWINDOW 4
ADD TPN IYAHTO80.M401SDRCSDR
ADD TPN IYAHTO80.M401RQSDSDR
ADD TPN IYAHTO80.M401RQSVSVR
ADD TPN IYAHTO80.M401SDRCCVR
ADD TPN IYAHTO80.M401RQSVRQSTR
ADD TPN IYAHTO80.M401RQSDRQSTR

==
== Stingray (AIX) Partner LU
==
ADD PTNR-LU IYAHTO80.IYA3T995, SNANAME GBIBMIYA.IYA3T995, & PERIPHERAL-NODE NO, PARALLEL-SESSION-LU YES
ADD PTNR-MODE IYAHTO80.IYA3T995.LU62PS, MODENAME LU62PS, & DEFAULTMAXSESSION 8, DEFAULTMINCONWINNER 4, & DEFAULTMINCONLOSER 3, MAXAUTOACT 1, RCVWINDOW 4, & DEFAULTMAXINRUSIZE 1024, DEFAULTMAXOUTRUSIZE 1024, & SENDWINDOW 4
ADD TPN IYAHTO80.MA02SDRCSDR
ADD TPN IYAHTO80.MA02RQSDSDR
ADD TPN IYAHTO80.MA02RQSVSVR
ADD TPN IYAHTO80.MA02SDRCCVR
ADD TPN IYAHTO80.MA02RQSVRQSTR
ADD TPN IYAHTO80.MA02RQSDRQSTR
== coop007 (OS/2) Partner LU ==

ADD PTNR-LU IYAHT080.IYAFT170, SNANAME GBIBMIYA.IYAFT170, &
PERIPHERAL-NODE NO, PARALLEL-SESSION-LU YES

ADD PTNR-MODE IYAHT080.IYAFT170.LU62PS, MODENAME LU62PS, &
DEFAULTMAXSESSION 8, DEFAULTMINCONWINNER 4, &
DEFAULTMINCONLOSER 3, MAXAUTOACT 1, RCVWINDOW 4, &
DEFAULTMAXINRUSIZE 1024, DEFAULTMAXOUTRUSIZE 1024, &
SENDWINDOW 4

ADD TPN IYAHT080.M002SDRCSDR
ADD TPN IYAHT080.M002RQSDS3DR
ADD TPN IYAHT080.M002RQSVS3VR
ADD TPN IYAHT080.M002SDRCRCVR
ADD TPN IYAHT080.M002RQSVSRQSTR
ADD TPN IYAHT080.M002RQSDRQSTR

== MQ-Portuguese (OS/2) Partner LU ==

ADD PTNR-LU IYAHT080.IYAHT090, SNANAME GBIBMIYA.IYAHT090, &
PERIPHERAL-NODE NO, PARALLEL-SESSION-LU YES

ADD PTNR-MODE IYAHT080.IYAHT090.LU62PS, MODENAME LU62PS, &
DEFAULTMAXSESSION 8, DEFAULTMINCONWINNER 4, &
DEFAULTMINCONLOSER 3, MAXAUTOACT 1, RCVWINDOW 4, &
DEFAULTMAXINRUSIZE 1024, DEFAULTMAXOUTRUSIZE 1024, &
SENDWINDOW 4

ADD TPN IYAHT080.M002SDRCSDR
ADD TPN IYAHT080.M002RQSDS3DR
ADD TPN IYAHT080.M002RQSVS3VR
ADD TPN IYAHT080.M002SDRCRCVR
ADD TPN IYAHT080.M002RQSVSRQSTR
ADD TPN IYAHT080.M002RQSDRQSTR

== finnr100 (NT) Partner LU ==

ADD PTNR-LU IYAHT080.IYAFT080, SNANAME GBIBMIYA.IYAFT080, &
PERIPHERAL-NODE NO, PARALLEL-SESSION-LU YES

ADD PTNR-MODE IYAHT080.IYAFT080.LU62PS, MODENAME LU62PS, &
DEFAULTMAXSESSION 8, DEFAULTMINCONWINNER 4, &
DEFAULTMINCONLOSER 3, MAXAUTOACT 1, RCVWINDOW 4, &
DEFAULTMAXINRUSIZE 1024, DEFAULTMAXOUTRUSIZE 1024, &
SENDWINDOW 4

ADD TPN IYAHT080.MN01SDRCSDR
ADD TPN IYAHT080.MN01RQSDS3DR
ADD TPN IYAHT080.MN01RQSVS3VR
ADD TPN IYAHT080.MN01SDRCRCVR
ADD TPN IYAHT080.MN01RQSVSRQSTR
ADD TPN IYAHT080.MN01RQSDRQSTR
Communications examples

== MVS CICS Partner LU ==

ADD PTNR-LU IYAHT080.IYCMVM03, SNANAME GBIBMIYA.IYCMVM03, &
PERIPHERAL-NODE NO, PARALLEL-SESSION-LU YES

ADD PTNR-MODE IYAHT080.IYCMVM03.LU62PS, MODENAME LU62PS, &
DEFAULTMAXSESSION 8, DEFAULTMINCONWINNER 4, &
DEFAULTMINCONLOSER 3, MAXAUTOACT 1, RCVWINDOW 4, &
DEFAULTMAXINRUSIZE 1024, DEFAULTMAXOUTRUSIZE 1024, &
SENDWINDOW 4

ADD TPN IYAHT080.VM03SDRCSDR
ADD TPN IYAHT080.VM03RQSDSDR
ADD TPN IYAHT080.VM03RQSVSVR
ADD TPN IYAHT080.VM03SDRCRCVR
ADD TPN IYAHT080.VM03RQSVRQSTR
ADD TPN IYAHT080.VM03RQSDRQSTR

== z/OS Non CICS Partner LU ==

ADD PTNR-LU IYAHT080.IYCNVM03, SNANAME GBIBMIYA.IYCNVM03, &
PERIPHERAL-NODE NO, PARALLEL-SESSION-LU YES

ADD PTNR-MODE IYAHT080.IYCNVM03.LU62PS, MODENAME LU62PS, &
DEFAULTMAXSESSION 8, DEFAULTMINCONWINNER 4, &
DEFAULTMINCONLOSER 3, MAXAUTOACT 1, RCVWINDOW 4, &
DEFAULTMAXINRUSIZE 1024, DEFAULTMAXOUTRUSIZE 1024, &
SENDWINDOW 4

ADD TPN IYAHT080.VM03NCMSDRCSDR
ADD TPN IYAHT080.VM03NCRQSDSDR
ADD TPN IYAHT080.VM03NCRQSVSVR
ADD TPN IYAHT080.VM03NCMSDRCRCVR
ADD TPN IYAHT080.VM03NCRQSVRQSTR
ADD TPN IYAHT080.VM03NCRQSDRQSTR

== VSE Partner LU ==

ADD PTNR-LU IYAHT080.IYZMZSI2, SNANAME GBIBMIYA.IYZMZSI2, &
PERIPHERAL-NODE NO, PARALLEL-SESSION-LU YES

ADD PTNR-MODE IYAHT080.IYZMZSI2.LU62PS, MODENAME LU62PS, &
DEFAULTMAXSESSION 8, DEFAULTMINCONWINNER 4, &
DEFAULTMINCONLOSER 3, MAXAUTOACT 1, RCVWINDOW 4, &
DEFAULTMAXINRUSIZE 1024, DEFAULTMAXOUTRUSIZE 1024, &
SENDWINDOW 4

== Start the LUs ==

START LU IYAHT080, SUB ALL
START TPN *
Channel definitions
Here are some example WebSphere MQ channel definitions that support the SNAX configuration:

- A sender channel to WebSphere MQ on z/OS (not using CICS):
  ```
  DEFINE CHANNEL(MT01.VM03.SDRC.0002) CHLTYPE(SDR) +
  TRPTYPE(LU62) +
  SEQWRAP(9999999) MAXMSGL(2048) +
  XMITQ('VM03NCM.TQ.SDRC.0001') +
  CONNAME('$BP01.IYAHT080.IYCNVM03') +
  MODENAME('LU62PS') TPNAME(DUMMY)
  ```

- A receiver channel from WebSphere MQ on z/OS:
  ```
  DEFINE CHANNEL(VM03.MT01.SDRC.0002) CHLTYPE(RCVR) +
  TRPTYPE(LU62) REPLACE DESCR('Receiver channel from VM03NCM') +
  SEQWRAP(9999999) +
  MAXMSGL(2048) AUTOSTART(ENABLED) +
  CONNAME('$BP01.IYAHT080') TPNAME(VM03NCMSDRCRCVR)
  ```

- A server channel to WebSphere MQ on z/OS which is capable of initiating a conversation, or being initiated by a remote requester channel:
  ```
  DEFINE CHANNEL(MT01.VM03.RQSV.0002) CHLTYPE(SVR) +
  TRPTYPE(LU62) +
  SEQWRAP(9999999) MAXMSGL(2048) +
  XMITQ('VM03NCM.TQ.RQSV.0001') +
  CONNAME('$BP01.IYAHT080.IYCNVM03') +
  MODENAME('LU62PS') TPNAME(VM03NCMRQSVSVR.DUMMY) +
  AUTOSTART(ENABLED)
  ```

  where DUMMY is the TPNAME the z/OS queue manager is listening on.

ICE communications example
There are two stages in configuring ICE for WebSphere MQ:
1. The ICE process itself must be configured.
2. Line ($ICE01, in the following example) and SNA information must be input to the ICE process.

Configuring the ICE process
Here is an example ICE process configuration. This configuration is located by default in a file called GOICE:
Communications examples

?tacl macro
clear all
param backupcpu 1
param cinittimer 120
param collector $0
param config icectl
param idblk 05d
param idnum 312FF
param cpname IYAHRO0C
param datapages 64
param dynamicrlu yes
param genesis $gen
param maxrcv 4096
param loglevel info
param netname GBIBMIYA
param password xxxxxxxxxxxxxxxxxxx
param retrysl 5
param secuserid super.super
param startup ½
param timer1 20
param timer2 300
param usstable default
run $system.ice.ice/name $ICE,nowait,cpu 0,pri 180,highpin off/

Note: The password param has been replaced by xxxxxxxxxxxxxxxxxxxx.

Defining the line and APC information
Once the ICE process has been started with this configuration, the following
information is input to the ICE process using the Node Operator Facility (NOF**).
This example defines a line called $ICE01 running on the token ring port
$CHAMB.#ICE:

==
== ICE definitions for PU IYAHRO0C.
== Local LU for this PU is IYAH0T0C0.
==
ALLOW ERRORS
OPEN $ICE
ABORT LINE $ICE01, SUB ALL
DELETE LINE $ICE01, SUB ALL

==
== ADD TOKEN RING LINE
==
ADD LINE $ICE01, TNDM $CHAMB.#ICE, &
   IDBLK %H05D, &
   PROTOCOL TOKENRING, WRITEBUFFERSIZE 8192
==
== ADD PU OBJECT
==
ADD PU IYAHRO0C, LINE $ICE01, MULTIROUTE YES, &
   DMAC 400045121088, DSAP %H04, &
   NETNAME GBIBMIYA, IDNUM %H312FF, IDBLK %H05D, &
   RCPNAME GBIBMIYA.IYACDRM, SSAP %H08

==
== Add Local APPL Object
==
DELETE APPL IYAHT0C
ADD APPL IYAHT0C, ALIAS IYAHT0C, LLU IYAHT0C, PROTOCOL CPIC, &
   OPENNAME #IYAHT0C

==
== Add Mode LU62PS
==
DELETE MODE LU62PS
ADD MODE LU62PS, MAXSESS 8, MINCONWIN 4, MINCONLOS 3

==
== Add Partner LU Objects
==

== spinach (HP) ==
ABORT RLU IYABT0F0
DELETE RLU IYABT0F0
ADD RLU IYABT0F0, MODE LU62PS, PARSESS YES

== stingray (AIX) ==
ABORT RLU IYA3T995
DELETE RLU IYA3T995
ADD RLU IYA3T995, MODE LU62PS, PARSESS YES

== coop007 (OS/2) ==
ABORT RLU IYAF170
DELETE RLU IYAF170
ADD RLU IYAF170, MODE LU62PS, PARSESS YES

== MVS CICS ==
ABORT RLU IYCMVM03
DELETE RLU IYCMVM03
ADD RLU IYCMVM03, MODE LU62PS, PARSESS YES
Communications examples

== MVS Non-CICS
ABORT RLU IYCNVM03
DELETE RLU IYCNVM03
ADD RLU IYCNVM03, MODE LU62PS, PARSESS YES

== finnr100 (NT)
ABORT RLU IYAOFT080
DELETE RLU IYAOFT080
ADD RLU IYAOFT080, MODE LU62PS, PARSESS YES

== winas18 (AS400)
ABORT RLU IYAOFT110
DELETE RLU IYAOFT110
ADD RLU IYAOFT110, MODE LU62PS, PARSESS YES
ABORT RLU IYAOFT080
DELETE RLU IYAOFT080
ADD RLU IYAOFT080, MODE LU62PS, PARSESS YES

==
== START UP ICE LINE $ICE01 AND SUB DEVICE
==

START LINE $ICE01, SUB ALL

Note: In order for this configuration to work, the port #ICE must have been defined to the token ring line. For example, these commands could be entered into SCF:
add port $chamb.#ice, type tr8025, address %H08
start port $chamb.#ice

where $chamb is a token-ring controller, and the SAP of the port is %08.

Channel definitions for ICE
Here are some WebSphere MQ channel definitions that would support this ICE configuration:

- A sender channel to WebSphere MQ on z/OS (not using CICS):
  
  DEFINE CHANNEL(MT01.VM03.SDRC.ICE) CHLTYPE(SDR) +
  TRPTYPE(LU62) +
  SEQWRAP(9999999) MAXMSGL(2048) +
  XMITQ('VM03NCM.TQ.SDRC.ICE') +
  CONNAME('ICE.#IYAHT0C.IYAHT0C0.IYCNVM03') +
  MODENAME('LU62PS') TPNAME(DUMMY)

- A receiver channel from WebSphere MQ on z/OS:
  
  DEFINE CHANNEL(VM03.MT01.SDRC.ICE) CHLTYPE(RCVR) +
  TRPTYPE(LU62) REPLACE DESCR('Receiver channel from VM03NCM') +
  SEQWRAP(9999999) +
  MAXMSGL(2048) AUTOSTART(ENABLED) +
  CONNAME('ICE.#IYAHT0C.IYAHT0C0') TPNAME(VM03NCMSDRCRCVR)

- A server channel to WebSphere MQ on z/OS that is capable of initiating a conversation, or being initiated by a remote requester channel:
  
  DEFINE CHANNEL(MT01.VM03.RQSV.ICE) CHLTYPE(SVR) +
  TRPTYPE(LU62) +
  SEQWRAP(9999999) MAXMSGL(2048) +
where _DUMMY_ is the TPNAME the MVS queue manager is listening on.

**TCP/IP communications example**

This example shows how to establish communications with a remote WebSphere MQ system over TCP/IP.

**TCPConfig stanza in QMINI**

The QMINI file must contain an appropriate TCPConfig stanza. For example:

```
TCPConfig:
    TCPPort=1414
    TCPNumListenerPorts=1
    TCPListenerPort=1996
    TCPKeepAlive=1
```

The TCPPort value is the default outbound port for channels without a port value in the CONNAME field. TCPListenerPort identifies the port on which the TCP listener will listen.

A queue manager can have multiple TCP/IP listeners. If this is the case, the QMINI file must have a TCPListenerPort entry for each listener, and TCPNumListenerPort must be updated to match. For example, the TCPConfig stanza above would be changed as follows:

```
TCPConfig:
    TCPPort=1414
    TCPNumListenerPorts=2
    TCPListenerPort=1997
    TCPKeepAlive=1
```

**Defining a TCP sender channel**

A TCP sender channel must be defined. In this example, the queue manager is MH01 on a host called SPINACH:

```
DEFINE CHANNEL(MT01_MH01_SDRC_0001) CHLTYPE(SDR) +
    TRPTYPE(TCP) +
    SEQWRAP(9999999) MAXMSGL(4194304) +
    XMITQ('MH01_TQ_SDRC_0001') +
    CONNAME('SPINACH.HURSLEY.IBM.COM(2000)')
```

This channel would try to attach to a TCP/IP port number 2000 on the host SPINACH.

The following example shows a TCP/IP sender channel definition for a queue manager MH01 on the host SPINACH using the default outbound TCP/IP port:

```
DEFINE CHANNEL(MT01_MH01_SDRC_0001) CHLTYPE(SDR) +
    TRPTYPE(TCP) +
    SEQWRAP(9999999) MAXMSGL(4194304) +
    XMITQ('MH01_TQ_SDRC_0001') +
    CONNAME('SPINACH.HURSLEY.IBM.COM')
```

No port number is specified in the CONNAME. Therefore, the value specified on the TCPPort entry in the QMINI file (1414) is used.

**Defining a TCP receiver channel**

An example TCP receiver channel:
A TCP receiver channel requires no CONNAME value, but a TCP listener must be running. There are two ways of starting a TCP listener. Either:

1. Go into the queue manager’s pathway using pathcom, and enter:
   ```
   start server mqs-tcplis00
   ```

   or

2. From the TACL prompt, enter
   ```
   runmqlsr -m QMgrName
   ```

   **Note:** If problems are encountered with the TACL from which the `runmqlsr` is running, the listener will be unable to access its home terminal and out file. `runmqlsr` is useful for testing, but you are recommended to use the listener from within the queue manager’s pathway as shown in step 1.

A TCP/IP listener, which will listen on the port defined in the QMINI file (in this example, 1996), is started.

**Note:** This port number can be overridden by the `-p port` flag on `runmqlsr`.

**Defining a TCP/IP sender channel on the remote system**

The sender channel definition on the remote system to connect to this receiver channel could look like:

```
DEFINE CHANNEL(MH01_MT01_SDRC_0001) CHLTYPE(SDR) +
    TRPTYPE(TCP) +
    XMITQ('MT01_TQ_SDRC_0001') +
    CONNAME('TANDEM.ISC.UK.IBM.COM(1996)')
```
Chapter 21. Message channel planning example for distributed platforms

This chapter provides a detailed example of how to connect two queue managers together so that messages can be sent between them. The example illustrates the preparations needed to allow an application using queue manager QM1 to put messages on a queue at queue manager QM2. An application running on QM2 can retrieve these messages, and send responses to a reply queue on QM1.

The example illustrates the use of TCP/IP connections. The example assumes that channels are to be triggered to start when the first message arrives on the transmission queue they are servicing. You must start the channel initiator in order for triggering to work.

This example uses SYSTEM.CHANNEL.INITQ as the initiation queue. This queue is already defined by WebSphere MQ. You can use a different initiation queue, but you will have to define it yourself and specify the name of the queue when you start the channel initiator.

What the example shows

The example shows the WebSphere MQ commands (MQSC) that you can use.

In all the examples, the MQSC commands are shown as they would appear in a file of commands, and as they would be typed at the command line. The two methods look identical, but, to issue a command at the command line, you must first type runmqsc, for the default queue manager, or runmqsc qmname where qmname is the name of the required queue manager. Then type any number of commands, as shown in the examples.

An alternative method is to create a file containing these commands. Any errors in the commands are then easy to correct. If you called your file mqsc.in then to run it on queue manager QMNAME use:

runmqsc QMNAME < mqsc.in > mqsc.out

You could verify the commands in your file before running it using:

runmqsc -v QMNAME < mqsc.in > mqsc.out

For portability, you should restrict the line length of your commands to 72 characters. Use a concatenation character to continue over more than one line. On Compaq NonStop Kernel use Ctrl-y to end the input at the command line, or enter the exit or quit command. On OS/2, Windows NT, or Compaq OpenVMS Alpha use Ctrl-z. On UNIX systems use Ctrl-d. Alternatively, on WebSphere MQ for AIX, iSeries, HP-UX, Linux, Solaris, and Windows, and MQSeries V5.1 for Compaq Tru64 UNIX, and OS/2 Warp, use the end command.

Figure 36 on page 324 shows the example scenario.
The example involves a payroll query application connected to queue manager QM1 that sends payroll query messages to a payroll processing application running on queue manager QM2. The payroll query application needs the replies to its queries sent back to QM1. The payroll query messages are sent from QM1 to QM2 on a sender-receiver channel called QM1.TO.QM2, and the reply messages are sent back from QM2 to QM1 on another sender-receiver channel called QM2.TO.QM1. Both of these channels are triggered to start as soon as they have a message to send to the other queue manager.

The payroll query application puts a query message to the remote queue “PAYROLL.QUERY” defined on QM1. This remote queue definition resolves to the local queue “PAYROLL” on QM2. In addition, the payroll query application specifies that the reply to the query is sent to the local queue “PAYROLL.REPLY” on QM1. The payroll processing application gets messages from the local queue “PAYROLL” on QM2, and sends the replies to wherever they are required; in this case, local queue “PAYROLL.REPLY” on QM1.

In the example definitions for TCP/IP, QM1 has a host address of 9.20.9.31 and is listening on port 1411, and QM2 has a host address of 9.20.9.32 and is listening on port 1412. The example assumes that these are already defined on your system and available for use.

The object definitions that need to be created on QM1 are:
- Remote queue definition, PAYROLL.QUERY
- Transmission queue definition, QM2 (default=remote queue manager name)
- Process definition, QM1.TO.QM2.PROCESS (not needed for WebSphere MQ for AIX, HP-UX, Solaris, and Windows, and MQSeries for Compaq Tru64 UNIX, and OS/2 Warp)
- Sender channel definition, QM1.TO.QM2
- Receiver channel definition, QM2.TO.QM1
- Reply-to queue definition, PAYROLL.REPLY

The object definitions that need to be created on QM2 are:
- Local queue definition, PAYROLL
- Transmission queue definition, QM1 (default=remote queue manager name)
- Process definition, QM2.TO.QM1.PROCESS (not needed for WebSphere MQ for AIX, HP-UX, Solaris, and Windows, and MQSeries for Compaq Tru64 UNIX, and OS/2 Warp)
- Sender channel definition, QM2.TO.QM1
Planning example for distributed platforms

- Receiver channel definition, QM1.TO.QM2

The connection details are supplied in the CONNAME attribute of the sender channel definitions.

You can see a diagram of the arrangement in Figure 36 on page 324.

Queue manager QM1 example

The following object definitions allow applications connected to queue manager QM1 to send request messages to a queue called PAYROLL on QM2, and to receive replies on a queue called PAYROLL.REPLY on QM1.

All the object definitions have been provided with the DESCR and REPLACE attributes. The other attributes supplied are the minimum required to make the example work. The attributes that are not supplied take the default values for queue manager QM1.

Run the following commands on queue manager QM1.

Remote queue definition

```
DEFINE QREMOTE(PAYROLL.QUERY) DESCR('Remote queue for QM2') REPLACE +
    PUT(ENABLED) XMITQ(QM2) RNAME(PAYROLL) RQMNAME(QM2)
```

**Note:** The remote queue definition is not a physical queue, but a means of directing messages to the transmission queue, QM2, so that they can be sent to queue manager QM2.

Transmission queue definition

```
DEFINE QLOCAL(QM2) DESCR('Transmission queue to QM2') REPLACE +
    USAGE(XMITQ) PUT(ENABLED) GET(ENABLED) TRIGGER TRIGTYPE(FIRST) +
    INITQ(SYSTEM.CHANNEL.INITQ) PROCESS(QM1.TO.QM2.PROCESS)
```

When the first message is put on this transmission queue, a trigger message is sent to the initiation queue, SYSTEM.CHANNEL.INITQ. The channel initiator gets the message from the initiation queue and starts the channel identified in the named process.

Process definition

```
DEFINE PROCESS(QM1.TO.QM2.PROCESS) DESCR('Process for starting channel') +
    REPLACE APPLTYPE(OS2) USERDATA(QM1.TO.QM2)
```

The channel initiator uses this process information to start channel QM1.TO.QM2. (This sample definition uses OS2 as the application type).

**Note:** For WebSphere MQ for AIX, HP-UX, Linux, Solaris, and Windows, and MQSeries V5.1 for Compaq Tru64 UNIX, and OS/2 Warp the need for a process definition can be eliminated by specifying the channel name in the TRIGDATA attribute of the transmission queue.

Sender channel definition

```
DEFINE CHANNEL(QM1.TO.QM2) CHLTYPE(SDR) TRPTYPE(TCP) +
    REPLACE DESCR('Sender channel to QM2') XMITQ(QM2) +
    CONNAME('9.20.9.32(1412)')
```

Receiver channel definition

```
DEFINE CHANNEL(QM2.TO.QM1) CHLTYPE(RCVR) TRPTYPE(TCP) +
    REPLACE DESCR('Receiver channel from QM2')
```

Reply-to queue definition
Planning example for distributed platforms

DEFINE QLOCAL(PAYROLL.REPLY) REPLACE PUT(ENABLED) GET(ENABLED) +
DESCR('Reply queue for replies to query messages sent to QM2')

The reply-to queue is defined as PUT(ENABLED). This ensures that reply
messages can be put to the queue. If the replies cannot be put to the
reply-to queue, they are sent to the dead-letter queue on QM1 or, if this
queue is not available, remain on transmission queue QM1 on queue
manager QM2. The queue has been defined as GET(ENABLED) to allow
the reply messages to be retrieved.

Queue manager QM2 example

The following object definitions allow applications connected to queue manager
QM2 to retrieve request messages from a local queue called PAYROLL, and to put
replies to these request messages to a queue called PAYROLL.REPLY on queue
manager QM1.

You do not need to provide a remote queue definition to enable the replies to be
returned to QM1. The message descriptor of the message retrieved from local
queue PAYROLL contains both the reply-to queue and the reply-to queue manager
names. Therefore, as long as QM2 can resolve the reply-to queue manager name to
that of a transmission queue on queue manager QM2, the reply message can be
sent. In this example, the reply-to queue manager name is QM1 and so queue
manager QM2 simply requires a transmission queue of the same name.

All the object definitions have been provided with the DESCR and REPLACE
attributes and are the minimum required to make the example work. The attributes
that are not supplied take the default values for queue manager QM2.

Run the following commands on queue manager QM2.

Local queue definition

DEFINE QLOCAL(PAYROLL) REPLACE PUT(ENABLED) GET(ENABLED) +
DESCR('Local queue for QM1 payroll details')

This queue is defined as PUT(ENABLED) and GET(ENABLED) for the
same reason as the reply-to queue definition on queue manager QM1.

Transmission queue definition

DEFINE QLOCAL(QM1) DESCR('Transmission queue to QM1') REPLACE +
USAGE(XMITQ) PUT(ENABLED) GET(ENABLED) TRIGGER TRIGTYPE(FIRST) +
INITQ(SYSTEM.CHANNEL.INITQ) PROCESS(QM2.TO.QM1.PROCESS)

When the first message is put on this transmission queue, a trigger
message is sent to the initiation queue, SYSTEM.CHANNEL.INITQ. The
channel initiator gets the message from the initiation queue and starts the
channel identified in the named process.

Process definition

DEFINE PROCESS(QM2.TO.QM1.PROCESS) DESCR('Process for starting channel') +
REPLACE APPLTYPE(OS2) USERDATA(QM2.TO.QM1)

The channel initiator uses this process information to start channel
QM2.TO.QM1. (This sample definition uses OS2 as the application type.)

Note: For WebSphere MQ for AIX, HP-UX, Linux, Solaris, and Windows,
and MQSeries V5.1 for Compaq Tru64 UNIX, and OS/2 Warp the
Planning example for distributed platforms

need for a process definition can be eliminated by specifying the channel name in the TRIGGERDATA attribute of the transmission queue.

Sender channel definition

```plaintext
DEFINE CHANNEL(QM2.TO.QM1) CHLTYPE(SDR) TRPTYPE(TCP) +
REPLACE DESCR('Sender channel to QM1') XMITQ(QM1) +
CONNAME('9.20.9.31(1411)')
```

Receiver channel definition

```plaintext
DEFINE CHANNEL(QM1.TO.QM2) CHLTYPE(RCVR) TRPTYPE(TCP) +
REPLACE DESCR('Receiver channel from QM1')
```

Running the example

Once these definitions have been created, you need to:

- Start the channel initiator on each queue manager.
- Start the INETD daemon for each queue manager. On OS/2, Windows, and Compaq NonStop Kernel, you can use the WebSphere MQ listener in place of INETD.

For information about starting the channel initiator and listener, see Chapter 10, "Setting up communication for OS/2 and Windows" on page 129 and Chapter 13, "Setting up communication in UNIX systems" on page 191.

Note: On OS/2 and Windows, you can also run the channel as a thread; see the WebSphere MQ Script (MQSC) Command Reference book for information about how to define a channel as a threaded channel.

Expanding this example

This simple example could be expanded with:

- The use of LU 6.2 communications for interconnection with CICS systems, and transaction processing.
- Adding more queue, process, and channel definitions to allow other applications to send messages between the two queue managers.
- Adding user-exit programs on the channels to allow for link encryption, security checking, or additional message processing.
- Using queue-manager aliases and reply-to queue aliases to understand more about how these can be used in the organization of your queue manager network.
Planning example for distributed platforms
Chapter 22. Example SINIX and DC/OSx configuration files

This chapter contains working examples of SNA LU 6.2 configuration files for SINIX and DC/OSx.

Notes:
1. The TCP/IP names for the SINIX machines involved are forties, which is an RM400, and bight, which is an RM200.
2. The name of the queue manager on forties is MP01, and the name of the queue manager on bight is MP02.
3. Both machines are running the SINIX-N operating system.
4. The LU names have a resemblance to the TCP/IP names.
5. The XIDs have been arbitrarily chosen to reflect the RM model numbers.
6. The machine rameses is a DC/OSx MIS-2ES/2 machine using the DC/OSx operating system. The configuration for rameses is different because the operating system SNA software on DC/OSx is different.
7. The name of the queue manager on rameses is MP04.

The preceding information can be summarized as follows:

<table>
<thead>
<tr>
<th>Machine name</th>
<th>Machine model</th>
<th>Operating system</th>
<th>Queue manager</th>
</tr>
</thead>
<tbody>
<tr>
<td>forties</td>
<td>RM400</td>
<td>SINIX-N</td>
<td>MP01</td>
</tr>
<tr>
<td>bight</td>
<td>RM200</td>
<td>SINIX-N</td>
<td>MP02</td>
</tr>
<tr>
<td>rameses</td>
<td>MIS-2ES/2</td>
<td>DC/OSx</td>
<td>MP04</td>
</tr>
</tbody>
</table>

You should use these examples as a basis for your system. You need to generate configuration files that are appropriate to your SNA network.

For a further description on the contents of KOGS files and Transit (SINIX LU6.2) setup, see the Transit SINIX Version 3.2 Administration of Transit manual.

The KOGS files can be found in the directory /opt/lib/transit/KOGS.

“Working configuration files for Pyramid DC/OSx” on page 332 shows example working configuration files from the DC/OSx machine rameses. The file is /etc/opt/lu62/cpic_cfg. For further information on the format of this file see the Pyramid Technology publications OpenNet LU 6.2, System Administrator’s Guide, and OpenNet SNA Engine, System Administrator’s Guide.

“Output of dbd command” on page 332 is the output of the dbd command on cfg.ncpram, which is a binary configuration file created by the cm command.
SINIX and DC/OSx configuration

Configuration file on bight

* Transit config file for bight (RM200).
* Versionen und Korrekturstaende
* TRANSIT-SERVER V 3.3 confnuc.h K1
* SNA_Kgen K1

XLINK forties,
   ACT = AUTO,
   TYP = LAN,
   XID = 00000400,
   CPNAME = CP.FORTIES,
   CONFSTR = /opt/lib/llc2/conf.str,
   DEVICE = tr0,
   SSAP = 04

XPU pbight,
   TYP = PEER,
   CONNECT = AUTO,
   DISCNT = AUTO,
   LINK = forties,
   NVSCONNECT = PARTNER,
   MAXDATA = 1033,
   XID = 00000200,
   CPNAME = CP.BIGHT,
   ROLE = NEG,
   PAUSE = 3,
   RETRIES = 10,
   DMAC = 000F01626436,
   DSAP = 04,
   RWINDOW = 7

XLU forties,
   TYP = 6,
   PUCONNECT = APHSTART,
   CTYP = PUBLIC,
   SESS-LMT = 130,
   SESS-CTR = IND,
   NETNAME = SNI.FORTIES,
   PAIR = bight MODE1

XRLU bight,
   NETNAME = SNI.BIGHT,
   PU = pbight

XMODE MODE1,
   SESS-MAX = 13,
   SESS-LOS = 6,
   SESS-WIN = 7,
   SESS-AUTO = 7,
   SRU-MAX = 87,
   RRU-MAX = 87,
   PAC-SEND = 0,
   PAC-RCV = 0

XSYMDEST sendMP02,
   RLU = bight,
   MODE = MODE1,
   TP = recvMP02,
   TP-TYP = USER,
   SEC-TYP = NONE

XTP recvMP01,
   UID = guenther,
   TYP = USER,
   PATH = /home/guenther/recvMP01.sh,
   SECURE = NO

XEND
Configuration file on forties

* Transit config file for forties (RM 400).
* Versionen und Korrekturstände
  * TRANSIT-SERVER V 3.3 confnuc.h K1
  * SNA_Kgen K1

```plaintext
XLINK lbight,
  ACT = AUTO,
  TYP = LAN,
  XID = 00000200,
  CPNAME = CP.BIGHT,
  CONFSTR = /opt/lib/llc2/conf.str,
  DEVICE = tr0,
  SSAP = 04

XPU pforties,
  TYP = PEER,
  CONNECT = AUTO,
  DISCNT = AUTO,
  LINK = lbight,
  NVSCONNECT = PARTNER,
  MAXDATA = 1033,
  XID = 00000400,
  CPNAME = CP.FORTIES,
  ROLE = NEG,
  PAUSE = 3,
  RETRIES = 10,
  DMAcC = 00006f106935,
  DSAP = 04,
  RWINDOW = 7

XLU bight,
  TYP = 6,
  PUCONNECT = APHSTART,
  CTYP = PUBLIC,
  SESS-LMT = 15,
  SESS-CTR = IND,
  NETNAME = SNI.BIGHT,
  PAIR = forties MODE1

XRLU forties,
  NETNAME = SNI.FORTIES,
  PU = pforties

XMODE MODE1,
  SESS-MAX = 13,
  SESS-LOS = 7,
  SESS-WIN = 6,
  SESS-AUTO = 6,
  SRU-MAX = 87,
  RRU-MAX = 87,
  PAC-SEND = 0,
  PAC-RCV = 0

XSYMDEST sendMP01,
  RLU = forties,
  MODE = MODE1,
  TP = recvMP01,
  TP-TYP = USER,
  SEC-TYP = NONE

XTP recvMP02,
  UID = guenther,
  TYP = USER,
  PATH = /home/guenther/recvMP02.sh,
  SECURE = NO

XEND
```
Working configuration files for Pyramid DC/OSx

# This is the side information file for CPI-C.
# The default file name is /etc/opt/lu62/cpic_cfg, use set environment variable CPIC_CFG to change the default.
#
# The lines starting with # are for comments; no blank lines are allowed.
# The format of each line is "1 2 3 4 5 6 7 8 9" all in one line.
# 1 - symbolic destination name
# 2 - local LU name (locally known name)
# 3 - remote LU name (locally known name)
# 4 - mode name
# 5 - remote TP name
# 6 - trace flag (1 if you want the trace on, 0 otherwise)
# 7 - security type (0 for none, 2 for program)
# 8 - user id (omit if security type is 0)
# 9 - password (omit if security type is 0)
#
# The following are some examples:
#
#sendMP02 LRAMESES BIGHT MODE1 recvMP02 1 0
#sendMP02 IYAF1F00 IYAF0000 LU62PS recvMP02 1 0
#sendMP03 IYAF1F00 IYAF0100 LU62PS recvMP03 1 0
#sendMP01 IYAF1F00 IYAF1200 LU62PS recvMP01 1 0
#sdEH01rc IYAF1F00 IYAF0B00 LU62PS MD06RCV 1 0
#sdEH01sv IYAF1F00 IYAF0B00 LU62PS MD06SRV 1 0
#sendVM01 IYAF1F00 IYAF0100 LU62PS INTCRS6A 1 0
#sendVM02 IYAF1F00 IYCMV002 LU62PS DUMMY 1 0
#sndvM1rc IYAF1F00 IYCMV002 LU62PS CKRC 1 0
#sndvM2sv IYAF1F00 IYCMV002 LU62PS CKSD 1 0

Output of dbd command

**** COMMUNICATIONS MANAGER DATABASE ****
Database version number 80

SNA CONTROLLER
controller name: SNA
controller execute name: 'startsna62 -c 24'

62 MANAGER
62 manager name: LU62MGR
62 manager execute name: 'lu62mgr'

LOCAL PU
local pu name: IYAF1F0
controller name: SNA
non-specific type pu
unsolicited recfms is NOT supported
xid format (0/3): 3

LOCAL LU
fully qualified local lu name (hex): c7 c2 c9 c2 d4 c9 e8 c1 4b c9 e8 c1 c6 e3 f1 c6 f0
fully qualified local lu name (ebcdic): GBIBMIYA.IYAF1F0
locally known local lu name: IYAF1F0
local pu name: IYAF1F0
lu number at the pu: 1
lu6.2 type lu
62 manager name: LU62MGR
lu session limit: 100
share limit: 2
send window size: 7
LU configuration options:
is NOT the default lu
SINIX and DC/OSx configuration

will NOT terminate on disconnect
printer can NOT be used in system mode
independent LU on BF connections

REMOTE PU
remote pu name: CPPG

REMOTE LU
fully qualified remote lu name (hex): c7 c2 c9 c2 d4 c9 e8 c1 4b c9 e8 c1 c6 e3 f0 f0 f0
fully qualified remote lu name (ebcdic): GBIBMIYA.IYAFT000
locally known remote lu name: IYAFT000
fully qualified local lu name (hex): c7 c2 c9 c2 d4 c9 e8 c1 4b c9 e8 c1 c6 e3 f1 c6 f0
fully qualified local lu name (ebcdic): GBIBMIYA.IYAFT1F0
uninterpreted remote lu name (hex): c9 e8 c1 c6 e3 f0 f0 f0
uninterpreted remote lu name (ebcdic): IYAFT000
remote pu name: CPPG
session initiation requests are initiate or queue
parallel sessions supported
no security information accepted
lu-lu verification NOT required
lu-lu password not displayed for security reasons

REMOTE LU
fully qualified remote lu name (hex): c7 c2 c9 c2 d4 c9 e8 c1 4b c9 e8 c1 c6 e3 f0 f1 f0
fully qualified remote lu name (ebcdic): GBIBMIYA.IYAFT010
locally known remote lu name: IYAFT010
fully qualified local lu name (hex): c7 c2 c9 c2 d4 c9 e8 c1 4b c9 e8 c1 c6 e3 f1 c6 f0
fully qualified local lu name (ebcdic): GBIBMIYA.IYAFT1F0
uninterpreted remote lu name (hex): c9 e8 c1 c6 e3 f0 f1 f0
uninterpreted remote lu name (ebcdic): IYAFT010
remote pu name: CPPG
session initiation requests are initiate or queue
parallel sessions supported
no security information accepted
lu-lu verification NOT required
lu-lu password not displayed for security reasons

REMOTE LU
fully qualified remote lu name (hex): c7 c2 c9 c2 d4 c9 e8 c1 4b c9 e8 c1 c5 e3 f1 f2 f0
fully qualified remote lu name (ebcdic): GBIBMIYA.IYAE120
locally known remote lu name: IYAE120
fully qualified local lu name (hex): c7 c2 c9 c2 d4 c9 e8 c1 4b c9 e8 c1 c6 e3 f1 c6 f0
fully qualified local lu name (ebcdic): GBIBMIYA.IYAE1F0
uninterpreted remote lu name (hex): c9 e8 c1 c5 e3 f1 f2 f0
uninterpreted remote lu name (ebcdic): IYAE120
remote pu name: CPPG
session initiation requests are initiate or queue
parallel sessions supported
no security information accepted
lu-lu verification NOT required
lu-lu password not displayed for security reasons

MODE
mode name (hex): e2 d5 c1 e2 e5 c3 d4 c7
mode name (ebcdic): SNASVCMG
fully qualified local lu name (hex): c7 c2 c9 c2 d4 c9 e8 c1 4b c9 e8 c1 c6 e3 f1 c6 f0
fully qualified local lu name (ebcdic): GBIBMIYA.IYAFT1F0
fully qualified remote lu name (hex): c7 c2 c9 c2 d4 c9 e8 c1 4b c9 e8 c1 c6 e3 f0 f0 f0
fully qualified remote lu name (ebcdic): GBIBMIYA.IYAFT000
line class name: leased
send pacing window: 7
receive pacing window: 7
lower bound max RU size, send: 128
upper bound max RU size, send: 896
lower bound max RU size, receive: 128
upper bound max RU size, receive: 896
synchronization level of none or confirm
either lu may attempt to reinitiate the session
cryptography not supported
contention-winner automatic initiation limit: 1

MODE
SINIX and DC/OSx configuration

mode name (hex): d3 e4 f6 f2 d7 e2
mode name (ebcdic): LU62PS
fully qualified local lu name (hex): c7 c2 c9 c2 d4 c9 e8 c1 4b c9 e8 c1 c6 e3 f1 c6 f0
fully qualified local lu name (ebcdic): GBIBMIYA.IYAF010
fully qualified remote lu name (hex): c7 c2 c9 c2 d4 c9 e8 c1 4b c9 e8 c1 c6 e3 f0 f0 f0
fully qualified remote lu name (ebcdic): GBIBMIYA.IYAF000
line class name: leased
send pacing window: 7
receive pacing window: 7
lower bound max RU size, send: 128
upper bound max RU size, send: 896
lower bound max RU size, receive: 128
upper bound max RU size, receive: 896
synchronization level of none or confirm
either lu may attempt to reinitiate the session
cryptography not supported
contention-winner automatic initiation limit: 5

MODE

mode name (hex): e2 d5 c1 e2 e5 c3 d4 c7
mode name (ebcdic): SNASVCMG
fully qualified local lu name (hex): c7 c2 c9 c2 d4 c9 e8 c1 4b c9 e8 c1 c6 e3 f1 c6 f0
fully qualified local lu name (ebcdic): GBIBMIYA.IYAFITFO
fully qualified remote lu name (hex): c7 c2 c9 c2 d4 c9 e8 c1 4b c9 e8 c1 c6 e3 f0 f1 f0
fully qualified remote lu name (ebcdic): GBIBMIYA.IYAF010
line class name: leased
send pacing window: 7
receive pacing window: 7
lower bound max RU size, send: 128
upper bound max RU size, send: 896
lower bound max RU size, receive: 128
upper bound max RU size, receive: 896
synchronization level of none or confirm
either lu may attempt to reinitiate the session
cryptography not supported
contention-winner automatic initiation limit: 1

MODE

mode name (hex): d3 e4 f6 f2 d7 e2
mode name (ebcdic): LU62PS
fully qualified local lu name (hex): c7 c2 c9 c2 d4 c9 e8 c1 4b c9 e8 c1 c6 e3 f1 c6 f0
fully qualified local lu name (ebcdic): GBIBMIYA.IYAFITFO
fully qualified remote lu name (hex): c7 c2 c9 c2 d4 c9 e8 c1 4b c9 e8 c1 c6 e3 f0 f1 f0
fully qualified remote lu name (ebcdic): GBIBMIYA.IYAF010
line class name: leased
send pacing window: 7
receive pacing window: 7
lower bound max RU size, send: 128
upper bound max RU size, send: 896
lower bound max RU size, receive: 128
upper bound max RU size, receive: 896
synchronization level of none or confirm
either lu may attempt to reinitiate the session
cryptography not supported
contention-winner automatic initiation limit: 5

MODE

mode name (hex): e2 d5 c1 e2 e5 c3 d4 c7
mode name (ebcdic): SNASVCMG
fully qualified local lu name (hex): c7 c2 c9 c2 d4 c9 e8 c1 4b c9 e8 c1 c6 e3 f1 c6 f0
fully qualified local lu name (ebcdic): GBIBMIYA.IYAFITFO
fully qualified remote lu name (hex): c7 c2 c9 c2 d4 c9 e8 c1 4b c9 e8 c1 c5 e3 f1 f2 f0
fully qualified remote lu name (ebcdic): GBIBMIYA.IYAF120
line class name: leased
send pacing window: 7
receive pacing window: 7
lower bound max RU size, send: 128
upper bound max RU size, send: 896
lower bound max RU size, receive: 128
upper bound max RU size, receive: 896
synchronization level of none or confirm
either lu may attempt to reinitiate the session
cryptography not supported
contention-winner automatic initiation limit: 1

**MODE**

mode name (hex): d3 e4 f6 f2 d7 e2
mode name (ebcdic): LU62PS
fully qualified local lu name (hex): c7 c2 c9 c2 d4 c9 e8 c1 4b c9 e8 c1 e3 f1 e6 f0
fully qualified local lu name (ebcdic): GBIBMIYA.IYAFT1F0
fully qualified remote lu name (hex): c7 c2 c9 c2 d4 c9 e8 c1 4b c9 e8 c1 c5 e3 f1 f2 f0
fully qualified remote lu name (ebcdic): GBIBMIYA.IYAET120

line class name: leased
send pacing window: 7
receive pacing window: 7
lower bound max RU size, send: 128
upper bound max RU size, send: 896
lower bound max RU size, receive: 128
upper bound max RU size, receive: 896
synchronization level of none or confirm
either lu may attempt to reinitiate the session
cryptography not supported
contention-winner automatic initiation limit: 5

**TRANSACTION PROGRAM**

transaction program name (hex): 99 b5 83 a5 d4 d7 f0 f4
transaction program name (ebcdic): recvMP04
transaction program execute name: '/home/guenther/recvMP04.sh'

tp is enabled
tp supports basic conversations
tp supports mapped conversations
tp supports confirm synchronization
tp supports no synchronization
no verification is required
number of pip fields required: 0

privilege mask (hex): 0
(no privileges)

**TRANSACTION PROGRAM**

transaction program name (hex): 06 f1
transaction program name (ebcdic): ?1
transaction program execute name: '06f1'

tp is enabled
tp supports basic conversations
tp supports confirm synchronization
tp supports no synchronization
no verification is required
number of pip fields required: 0

privilege mask (hex): 82
(cnos - allocate_service_tp privileges)

**TOKEN RING COMMUNICATIONS MEDIA**

line name: LINE0
line number: 0
controller name: SNA
line class: leased

**LOCAL LINK STATION**

link station name: LYAFT1F0
pu name: IYAFT1F0
line name: LINE0
secondary station
LSAP address (in hex): 04
i-field size: 1033
Acknowledgement delay window size : 7
Acknowledgement delay timeout in tenth of seconds : 3
Retry count : 20
Retry timeout in seconds : 3
send xid block number: 0 5d
send xid id number: 3 0f 5c
send xid control vector:
SINIX and DC/OSx configuration

REMOTE LINK STATION
  link station name: LCPPG
  pu name: CPPG
  line name: LINE0
  primary station
  MAC address: 40 00 45 12 10 88
  LSAP address (in hex): 04
  i-field size: 1033
  Remote station type : BF
  send xid block number:
  send xid id number:
  send xid control vector:
Part 4. DQM in WebSphere MQ for z/OS

Chapter 23. Monitoring and controlling channels on z/OS .................................................. 341
   The DQM channel control function ................................................................. 341
   Using the panels and the commands .............................................................. 342
      Using the initial panel ................................ ..................................................... 342
   Managing your channels ................................................................................. 344
      Defining a channel ......................................................................................... 344
      Altering a channel definition ...................................................................... 345
      Displaying a channel definition .................................................................. 345
      Deleting a channel definition ..................................................................... 346
      Displaying information about DQM ............................................................. 346
      Starting a channel initiator ........................................................................ 346
      Stopping a channel initiator ........................................................................ 347
      Starting a channel listener .......................................................................... 348
      Stopping a channel listener ........................................................................ 349
      Starting a channel ........................................................................................ 349
      Starting a shared channel .......................................................................... 350
      Testing a channel .......................................................................................... 351
      Resetting message sequence numbers for a channel ......................... 351
      Resolving in-doubt messages on a channel .............................................. 352
      Stopping a channel ....................................................................................... 352
      Usage notes .................................................................................................. 353
      Displaying channel status ......................................................................... 354
      Displaying cluster channels ...................................................................... 355

Chapter 24. Preparing WebSphere MQ for z/OS .......................................................... 357
   Defining DQM requirements to WebSphere MQ .......................................... 357
   Defining WebSphere MQ objects ...................................................................... 357
      Transmission queues and triggering channels ........................................... 357
      Synchronization queue ............................................................................... 358
      Channel command queues ......................................................................... 358
      Starting the channel initiator ...................................................................... 359
      Stopping the channel initiator .................................................................... 359
   Other things to consider .................................................................................. 359
      Operator Messages ......................................................................................... 359
      Channel operation commands ...................................................................... 359
      Undelivered-message queue ........................................................................ 359
      Queues in use .................................................................................................. 360
      Security changes .......................................................................................... 360
      Communications stopped ............................................................................ 360
      TCP .................................................................................................................. 360
      LU6.2 .............................................................................................................. 360
   z/OS Automatic Restart Management (ARM) ................................................... 360

Chapter 25. Setting up communication for z/OS ...................................................... 361
   Deciding on a connection ............................................................................... 361
   Defining a TCP connection ............................................................................. 361
      Sending end .................................................................................................... 362
      Receiving on TCP .......................................................................................... 362
      Using the TCP listener backlog option ....................................................... 362
   Defining an LU6.2 connection ........................................................................ 363
   APPC/MVS setup .............................................................................................. 363
      Connecting to APPC/MVS (LU 6.2) .............................................................. 364
      Receiving on LU 6.2 ....................................................................................... 364

Chapter 26. Example configuration - IBM WebSphere MQ for z/OS ......................... 365
   Configuration parameters for an LU 6.2 connection ....................................... 365
      Configuration worksheet ............................................................................... 366
      Explanation of terms ..................................................................................... 367
      Establishing an LU 6.2 connection ............................................................... 368
      Defining yourself to the network ................................................................ 368
      Defining a connection to a partner .............................................................. 370
      Establishing a TCP connection .................................................................... 371
      What next? ..................................................................................................... 371
   WebSphere MQ for z/OS configuration ............................................................. 371
      Channel configuration .................................................................................. 372
         WebSphere MQ for z/OS sender-channel definitions using LU 6.2 ........ 374
         WebSphere MQ for z/OS receiver-channel definitions using LU 6.2 .... 374
         WebSphere MQ for z/OS sender-channel definitions using TCP .......... 374
         WebSphere MQ for z/OS receiver-channel definitions using TCP ....... 375

Chapter 27. Message channel planning example for z/OS ........................................ 377
   What the example shows ................................................................................ 377
      Queue manager QM1 example ..................................................................... 378
      Remote queue definition ................................................................................. 378
      Transmission queue definition ...................................................................... 378
      Sender channel definition ............................................................................. 379
      Receiver channel definition ......................................................................... 379
      Reply-to queue definition ............................................................................. 379
      Queue manager QM2 example ..................................................................... 379
      Local queue definition .................................................................................. 380
      Transmission queue definition ..................................................................... 380
      Sender channel definition ............................................................................. 380
      Receiver channel definition ......................................................................... 380
   Running the example ....................................................................................... 380
   Expanding this example ................................................................................... 380

Chapter 28. Monitoring and controlling channels in z/OS with CICS ......................... 383
   The DQM channel control function .................................................................. 383
   CICS regions ...................................................................................................... 384
   Starting DQM panels ....................................................................................... 384
   The Message Channel List panel .................................................................... 385
      Keyboard functions ......................................................................................... 385
         Function keys .............................................................................................. 385
         Enter key ....................................................................................................... 386
         Clear key ....................................................................................................... 386
         Unassigned keys and unavailable choices ................................................ 386
   Selecting a channel ........................................................................................... 386
   Working with channels ...................................................................................... 386
   Creating a channel ............................................................................................ 387
   Altering a channel .............................................................................................. 388
   Browsing a channel .......................................................................................... 388
   Renaming a channel .......................................................................................... 389
   Selected menu-bar choice ............................................................................... 389
DQM in WebSphere MQ for z/OS

Start ........................................ 389
Stop ......................................... 391
Resync ....................................... 393
Reset ......................................... 394
Resolve ....................................... 395
Display status .............................. 396
Display settings ............................ 397
Ping ........................................... 398
Exit ........................................... 399
Edit menu-bar choice ....................... 399
Copy ........................................... 399
Create ....................................... 400
Alter .......................................... 402
Delete ........................................ 402
Find .......................................... 402
View menu-bar choice ....................... 403
Help menu-bar choice ....................... 404
The channel definition panels .......... 404
Channel menu-bar choice .................. 405
Saving changes ............................. 405
Exit from the panel ........................ 405
Help menu-bar choice ....................... 405
Channel settings panel fields .......... 406
Details of sender channel settings panel 408
Details of receiver channel settings panel 409
Details of server channel settings panel . 410
Details of requester channel settings panel. 411

Chapter 29. Preparing WebSphere MQ for z/OS
when using CICS .......................... 413
Defining DQM requirements to WebSphere MQ 413
Defining WebSphere MQ objects .......... 413
Multiple message channels per transmission queue ................................. 414
Channel operation considerations .... 415

Chapter 30. Setting up CICS communication for
WebSphere MQ for z/OS .................... 417
Connecting CICS systems ............... 417
Communication between queue managers 417
Intersystem communication ............... 417
Defining an LU 6.2 connection .......... 418
Installing the connection .................. 419
Communications between CICS systems attached to one queue manager 419
Connection names for function shipping . 419

Chapter 31. Example configuration - IBM
WebSphere MQ for z/OS using CICS ........ 421
Configuration parameters for an LU 6.2 connection 421
Configuration worksheet .................. 422
Explanation of terms ....................... 423
Establishing an LU 6.2 connection using CICS .... 424
Defining a connection ....................... 424
Defining the sessions ....................... 425
Installing the new group definition .... 426
What next? ................................ 426
WebSphere MQ for z/OS using CICS configuration 426
Channel configuration ....................... 427
WebSphere MQ for z/OS sender-channel definitions using CICS ........ 428
WebSphere MQ for z/OS receiver-channel definitions using CICS ........ 428

Chapter 32. Message channel planning example
for z/OS using CICS ....................... 431

Chapter 33. Preparing WebSphere MQ for z/OS
for DQM with queue-sharing groups .... 439
Concepts ..................................... 439
Class of service ................................ 439
Generic interface ............................ 439
Components .................................. 439
Listeners ..................................... 439
Transmission queues and triggering .... 440
Triggering .................................... 440
Message channel agents ................... 440
Inbound ....................................... 441
Outbound ..................................... 441
Synchronization queue .................... 441
Benefits ....................................... 441
Load-balanced channel start ............. 441
Shared channel recovery ................. 442
Client channels ............................. 442
Clusters and queue-sharing groups .... 442
Channels and serialization ................ 443
Intra-group queuing ....................... 443

Chapter 34. Setting up communication for
WebSphere MQ for z/OS using queue-sharing
groups ........................................ 445
Deciding on a connection .................. 445
Defining a TCP connection ............... 445
Sending end .................................. 445
Receiving on TCP using a queue-sharing group 445
Defining an LU6.2 connection .......... 445
Connecting to APPC/MVS (LU 6.2) ....... 446
Receiving on LU 6.2 using a generic interface ....... 446

Chapter 35. Example configuration - WebSphere MQ for z/OS using queue-sharing groups .... 447
Configuration parameters for an LU 6.2 connection 447
Configuration worksheet .................. 448
Explanation of terms ....................... 449
Establishing an LU 6.2 connection into a queue-sharing group .......... 450
Defining yourself to the network using generic resources ............. 450
Defining a connection to a partner ........ 452
What next? ................................ 452
Establishing a TCP connection into a queue-sharing group .......... 453
Using WLM/DNS ............................. 453
What next? ................................ 453
WebSphere MQ for z/OS shared channel configuration .......... 454
Shared channel configuration ............ 454
WebSphere MQ for z/OS shared receiver-channel definitions using LU 6.2 . 455
WebSphere MQ for z/OS shared sender-channel definitions using LU 6.2 . 455
Chapter 36. Message channel planning example for z/OS using queue-sharing groups

What this example shows ........................................... 457
Queue-sharing group definitions ................................... 459
Shared objects .......................................................... 459
Group objects .......................................................... 459
Queue manager QM3 example ....................................... 459
Sender channel definition ........................................... 460
Remaining definitions ............................................... 460
Running the example ................................................ 460

Chapter 37. Intra-group queuing

Concepts ................................................................. 461
Intra-group queuing and the intra-group queuing agent ............... 461
Terminology ............................................................. 462
Intra-group queuing .................................................. 462
Shared transmission queue for use by intra-group queuing .......... 463
Intra-group queuing agent ........................................... 463
Benefits ................................................................. 463
Reduced system definitions .......................................... 463
Reduced system administration ...................................... 463
Improved performance ............................................... 463
Supports migration .................................................... 463
Transparent delivery of messages when multi-hopping between queue managers in a queue-sharing group ........ 464

Chapter 38. Example Configuration — WebSphere MQ for z/OS using intra-group queuing

Configuration 1 ......................................................... 475
Configuration 2 ......................................................... 476
Configuration 3 ......................................................... 476
Configuration 1 definitions ........................................... 477
On QMG1 .................................................................. 477
On QMG2 .................................................................. 478
On QMG3 .................................................................. 478
Configuration 2 definitions ........................................... 479
On QMG1 .................................................................. 479
On QMG2 .................................................................. 479
On QMG3 .................................................................. 480
Configuration 3 definitions ........................................... 480
On QMG1 .................................................................. 480
On QMG2 .................................................................. 480
On QMG3 .................................................................. 481
Running the example ................................................... 481
Expanding the example ................................................ 481

Important notice

Distributed queuing using CICS ISC is retained for compatibility with previous releases; there will be no further enhancements to this function. Therefore you are recommended to use the channel initiator for distributed queuing.

Part 4. DQM in WebSphere MQ for z/OS 339
DQM in WebSphere MQ for z/OS
Chapter 23. Monitoring and controlling channels on z/OS

Use the DQM commands and panels to create, monitor, and control the channels to remote queue managers. Each z/OS queue manager has a DQM program (the channel initiator) for controlling interconnections to remote queue managers using native z/OS facilities.

The implementation of these panels and commands on z/OS is integrated into the operations and control panels and the MQSC commands. No differentiation is made in the organization of these two sets of panels and commands.

The information in this chapter applies in all cases where the channel initiator is used for distributed queuing. It applies whether or not you are using queue-sharing groups, or intra-group queuing.

If you are using CICS for DQM, the information in this chapter does not apply. See Chapter 28, “Monitoring and controlling channels in z/OS with CICS” on page 383 for more information.

The DQM channel control function

The channel control function provides the administration and control of message channels between WebSphere MQ for z/OS and remote systems. See Figure 28 on page 58 for a conceptual picture.

The channel control function consists of panels, commands and programs, two synchronization queues, channel command queues, and the channel definitions. The following is a brief description of the components of the channel control function.

- The channel definitions are held as objects in page set zero or in DB2®, like other WebSphere MQ objects in z/OS.
- You use the operations and control panels or MQSC commands to:
  - Create, copy, display, alter, and delete channel definitions
  - Start and stop channel initiators and listeners
  - Start, stop, and ping channels, reset channel sequence numbers, and resolve in-doubt messages when links cannot be re-established
  - Display status information about channels
  - Display information about DQM

In particular, you can use the CSQINPX initialization input data set to issue your MQSC commands. This can be processed every time you start the channel initiator. See the WebSphere MQ for z/OS Concepts and Planning Guide for information about this.

- There are two queues (SYSTEM.CHANNEL.SYNCQ and SYSTEM.QSG.CHANNEL.SYNCQ) used for channel re-synchronization purposes. You should define these with INDXTYPE(MSGID) for performance reasons.
- The channel command queue (SYSTEM.CHANNEL.INITQ) is used to hold commands for channel initiators, channels, and listeners.
Channel control function

- The channel control function program runs in its own address space, separate from the queue manager, and comprises the channel initiator, listeners, MCAs, trigger monitor, and command handler.
- For queue-sharing groups and shared channels, see Chapter 33, “Preparing WebSphere MQ for z/OS for DQM with queue-sharing groups” on page 439.
- For Intra-group queuing, see Chapter 37, “Intra-group queuing” on page 461.

Using the panels and the commands

You can use either the MQSC commands or the operations and control panels to manage DQM. For information about the syntax of the MQSC commands, see the WebSphere MQ Script (MQSC) Command Reference book.

Using the initial panel

For an introduction to invoking the operations and control panels, using the function keys, and getting help, see the WebSphere MQ for z/OS System Administration Guide.

Note: To use the operations and control panels, you must have the correct security authorization; see the WebSphere MQ for z/OS System Setup Guide for information. Figure 37 shows the panel that is displayed when you start a panel session. The text after the panel explains the actions you should perform in this panel.

From this panel you can:
- Select the action you want to perform by typing in the appropriate number in the Action field.
- Specify the object type that you want to work with. Press F4 for a list of object types if you are not sure what they are.
- Display a list of objects of the type specified. Type in an asterisk (*) in the Name field and press Enter to display a list of objects (of the type specified) that have
already been defined on this subsystem. You can then select one or more objects
to work with in sequence. Figure 38 shows a list of channels produced in this
way.

- Specify the disposition in the queue-sharing group of the objects you want to
work with in the **Disposition** field. The disposition determines where the object
is kept and how the object behaves.

- Choose the local queue manager, or queue-sharing group to which you want to
connect in the **Connect name** field. If you want the commands to be issued on a
remote queue manager, choose either the **Target queue manager** field or the
**Action queue manager** field, depending upon whether the remote queue
manager is not or is a member of a queue-sharing group. If the remote queue
manager is not a member of a queue-sharing group, choose the **Target queue
manager** field. If the remote queue manager is a member of a queue-sharing
group, choose the **Action queue manager** field.

- Choose the wait time for responses to be received in the **Response wait time**
field.

```
List Channels - MQ25  Row 1 of 8

Type action codes. Then press Enter.
1=Display  2=Define Like  3=Alter  4=Manage  5=Perform
6=Start    7=Stop

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Disposition</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;&gt; * CHANNEL ALL</td>
<td>MQ25</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SYSTEM.DEF.CLNTCONN</td>
<td>CLNTCONN</td>
<td>QMGR MQ25</td>
<td></td>
</tr>
<tr>
<td>SYSTEM.DEF.CLUSSCVR</td>
<td>CLUSSCVR</td>
<td>QMGR MQ25 INACTIVE</td>
<td></td>
</tr>
<tr>
<td>SYSTEM.DEF.RECEIVER</td>
<td>RECEIVER</td>
<td>QMGR MQ25 INACTIVE</td>
<td></td>
</tr>
<tr>
<td>SYSTEM.DEF.REQUESTER</td>
<td>REQUESTER</td>
<td>QMGR MQ25 INACTIVE</td>
<td></td>
</tr>
<tr>
<td>SYSTEM.DEF.SENDER</td>
<td>SENDER</td>
<td>QMGR MQ25 INACTIVE</td>
<td></td>
</tr>
<tr>
<td>SYSTEM.DEF.SERVER</td>
<td>SERVER</td>
<td>QMGR MQ25 INACTIVE</td>
<td></td>
</tr>
<tr>
<td>SYSTEM.DEF.SVRCONN</td>
<td>SVRCONN</td>
<td>QMGR MQ25 INACTIVE</td>
<td></td>
</tr>
</tbody>
</table>

****** End of list *******
```

*Figure 38. Listing channels*
## Managing your channels

Table 31 lists the tasks that you can perform to manage your channels, channel initiators, and listeners. It also gives the name of the relevant MQSC command, and points to the page where each task is discussed.

**Table 31. Channel tasks**

<table>
<thead>
<tr>
<th>Task to be performed</th>
<th>MQSC command</th>
<th>See page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Define a channel</td>
<td>DEFINE CHANNEL</td>
<td>344</td>
</tr>
<tr>
<td>Alter a channel definition</td>
<td>ALTER CHANNEL</td>
<td>345</td>
</tr>
<tr>
<td>Display a channel definition</td>
<td>DISPLAY CHANNEL</td>
<td>345</td>
</tr>
<tr>
<td>Delete a channel definition</td>
<td>DELETE CHANNEL</td>
<td>345</td>
</tr>
<tr>
<td>Start a channel initiator</td>
<td>START CHINIT</td>
<td>346</td>
</tr>
<tr>
<td>Stop a channel initiator</td>
<td>STOP CHINIT</td>
<td>347</td>
</tr>
<tr>
<td>Display channel initiator information</td>
<td>DISPLAY DQM</td>
<td>346</td>
</tr>
<tr>
<td>Start a channel listener</td>
<td>START LISTENER</td>
<td>348</td>
</tr>
<tr>
<td>Stop a channel listener</td>
<td>STOP LISTENER</td>
<td>349</td>
</tr>
<tr>
<td>Start a channel</td>
<td>START CHANNEL</td>
<td>349</td>
</tr>
<tr>
<td>Test a channel</td>
<td>PING CHANNEL</td>
<td>351</td>
</tr>
<tr>
<td>Reset message sequence numbers for a channel</td>
<td>RESET CHANNEL</td>
<td>351</td>
</tr>
<tr>
<td>Resolve in-doubt messages on a channel</td>
<td>RESOLVE CHANNEL</td>
<td>352</td>
</tr>
<tr>
<td>Stop a channel</td>
<td>STOP CHANNEL</td>
<td>352</td>
</tr>
<tr>
<td>Display channel status</td>
<td>DISPLAY CHSTATUS</td>
<td>354</td>
</tr>
<tr>
<td>Display cluster channels</td>
<td>DISPLAY CLUSQMGR</td>
<td>355</td>
</tr>
</tbody>
</table>

### Defining a channel

To define a channel using the MQSC commands, use DEFINE CHANNEL.

Using the operations and control panels, starting from the initial panel, complete these fields and press Enter:

- **Field** | **Value**
  - Action: 2 (Define like)
  - Object type: channel type (for example SENDER) or CHANNEL
  - Name
  - Disposition: The location of the new object.

You are presented with some panels to complete with information about the name and attributes you want for the channel you are defining. They are initialized with the default attribute values. Change any you want before pressing Enter.

**Note:** If you entered CHANNEL in the **object type** field, you are presented with the Select a Valid Channel Type panel first.

If you want to define a channel with the same attributes as an existing channel, put the name of the channel you want to copy in the **Name** field on the initial panel. The panels will be initialized with the attributes of the existing object.
For information about the channel attributes, see Chapter 6, “Channel attributes” on page 77.

Notes:

1. If you are using distributed queuing with CICS as well, don’t use any of the same channel names.

2. You are strongly recommended to name all the channels in your network uniquely. As shown in Table 1 on page 30, including the source and target queue manager names in the channel name is a good way to do this.

Altering a channel definition

To alter a channel definition using the MQSC commands, use ALTER CHANNEL.

Using the operations and control panels, starting from the initial panel, complete these fields and press Enter:

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Action</td>
<td>3 (Alter)</td>
</tr>
<tr>
<td>Object type</td>
<td>channel type (for example SENDER) or CHANNEL</td>
</tr>
<tr>
<td>Name</td>
<td>CHANNEL.TO.ALTER</td>
</tr>
<tr>
<td>Disposition</td>
<td>The location of the stored object.</td>
</tr>
</tbody>
</table>

You are presented with some panels containing information about the current attributes of the channel. Change any of the unprotected fields that you want by overtyping the new value, and then press Enter to change the channel definition.

For information about the channel attributes, see Chapter 6, “Channel attributes” on page 77.

Displaying a channel definition

To display a channel definition using the MQSC commands, use DISPLAY CHANNEL.

Using the operations and control panels, starting from the initial panel, complete these fields and press Enter:

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Action</td>
<td>1 (Display)</td>
</tr>
<tr>
<td>Object type</td>
<td>channel type (for example SENDER) or CHANNEL</td>
</tr>
<tr>
<td>Name</td>
<td>CHANNEL.TO.DISPLAY</td>
</tr>
<tr>
<td>Disposition</td>
<td>The location of the object.</td>
</tr>
</tbody>
</table>

You are presented with some panels displaying information about the current attributes of the channel.

For information about the channel attributes, see Chapter 6, “Channel attributes” on page 77.

Deleting a channel definition

To delete a channel definition using the MQSC commands, use DELETE CHANNEL.
Deleting a channel definition

Using the operations and control panels, starting from the initial panel, complete these fields and press Enter:

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Action</td>
<td>4 (Manage)</td>
</tr>
<tr>
<td>Object type</td>
<td>channel type (for example SENDER) or CHANNEL</td>
</tr>
<tr>
<td>Name</td>
<td>CHANNEL.TO.DELETE</td>
</tr>
<tr>
<td>Disposition</td>
<td>The location of the object.</td>
</tr>
</tbody>
</table>

You are presented with another panel. Select function type 1 on this panel.

Press Enter to delete the channel definition; you will be asked to confirm that you want to delete the channel definition by pressing Enter again.

**Note:** The channel initiator has to be running before a channel definition can be deleted (except for client-connection channels).

Displaying information about DQM

To display information about the channel initiator using the MQSC commands, use DISPLAY DQM.

Using the operations and control panels, starting from the initial panel, complete these fields and press Enter:

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Action</td>
<td>1 (Display)</td>
</tr>
<tr>
<td>Object type</td>
<td>SYSTEM</td>
</tr>
<tr>
<td>Name</td>
<td>Blank</td>
</tr>
</tbody>
</table>

You are presented with another panel. Select function type 1 on this panel.

**Notes:**
1. Displaying distributed queuing information may take some time if you have lots of channels.
2. The channel initiator has to be running before you can display information about distributed queuing.

Starting a channel initiator

To start a channel initiator using the MQSC commands, use START CHINIT.

Using the operations and control panels, starting from the initial panel, complete these fields and press Enter:

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Action</td>
<td>6 (Start)</td>
</tr>
<tr>
<td>Object type</td>
<td>SYSTEM</td>
</tr>
<tr>
<td>Name</td>
<td>Blank</td>
</tr>
</tbody>
</table>

The Start a System Function panel is displayed. The text following the panel below explains what action you should take.
Starting a channel initiator

Select function type 1 (channel initiator), and press Enter. The channel initiator parameter module name defaults to CSQXPARM. If you want to use a different parameter module, enter the name on the panel.

Note: If you are using Interlink TCP, this must be started before you start the channel initiator. If you are using IBM TCP, you can start the channel initiator first but, unless you are using OE sockets, you will need to restart the channel initiator after you have started TCP, in order to establish communication. If you are using LU 6.2, this can be started before or after the channel initiator.

Stopping a channel initiator

To stop a channel initiator using the MQSC commands, use STOP CHINIT.

Using the operations and control panels, starting from the initial panel, complete these fields and press Enter:

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Action</td>
<td>7 (Stop)</td>
</tr>
<tr>
<td>Object type</td>
<td>SYSTEM</td>
</tr>
<tr>
<td>Name</td>
<td>Blank</td>
</tr>
</tbody>
</table>

The Stop a System Function panel is displayed. The text following the panel explains how you should use this panel.
### Stopping a channel initiator

Select function type 1 (channel initiator) and press Enter.

The channel initiator will wait for all running channels to stop in quiesce mode before it stops.

**Note:** If some of the channels are receiver or requester channels that are running but not active, a stop request issued to either the receiver’s or sender’s channel initiator will cause it to stop immediately.

However, if messages are flowing, the channel initiator waits for the current batch of messages to complete before it stops.

#### Starting a channel listener

To start a channel listener using the MQSC commands, use START LISTENER.

Using the operations and control panels, starting from the initial panel, complete these fields and press Enter:

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Action</td>
<td>6 (Start)</td>
</tr>
<tr>
<td>Object type</td>
<td>SYSTEM</td>
</tr>
<tr>
<td>Name</td>
<td>Blank</td>
</tr>
</tbody>
</table>

The Start a System Function panel is displayed (see Figure 39 on page 347).

Select function type 2 (channel listener). Select Inbound disposition. Select Transport type. If the Transport type is L, select LU name. If the Transport type is T, select Port number and (optionally) IP address. Press Enter.

**Note:** For the TCP/IP listener, you can start multiple combinations of Port and IP address.
Stopping a channel listener

To stop a channel listener using the MQSC commands, use STOP LISTENER.

Using the operations and control panels, starting from the initial panel, complete these fields and press Enter:

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Action</td>
<td>7 (Stop)</td>
</tr>
<tr>
<td>Object type</td>
<td>SYSTEM</td>
</tr>
<tr>
<td>Name</td>
<td>Blank</td>
</tr>
</tbody>
</table>

The Stop a System Function panel is displayed (see Figure 40 on page 348).

Select function type 2 (channel listener). Select Inbound disposition. Select Transport type. If the transport type is 'T', select Port number and (optionally) IP address. Press Enter.

**Note:** For a TCP/IP listener, you can stop specific combinations of Port and IP address, or you can stop all combinations.

Starting a channel

To start a channel using the MQSC commands, use START CHANNEL.

Using the operations and control panels, starting from the initial panel, complete these fields and press Enter:

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Action</td>
<td>6 (Start)</td>
</tr>
<tr>
<td>Object type</td>
<td>channel type (for example SENDER) or CHANNEL</td>
</tr>
<tr>
<td>Name</td>
<td>CHANNEL.TO.USE</td>
</tr>
<tr>
<td>Disposition</td>
<td>The disposition of the object.</td>
</tr>
</tbody>
</table>

The Start a Channel panel is displayed. The text following the panel explains how to use the panel.
Starting a channel

Select the disposition of the channel instance and on which queue manager it is to be started.

Press Enter to start the channel.

Starting a shared channel
To start a shared channel, and keep it on a nominated channel initiator, use disposition = S (on the START CHANNEL command, specify CHLDISP(FIXSHARED)).

When you start a channel in this way, the following rules apply to that channel:

• You can stop the channel from any queue manager in the queue-sharing group. You can do this even if the channel initiator on which it was started is not running at the time you issue the stop-channel request. When the channel has stopped, you can restart it by specifying disposition = S (CHLDISP(FIXSHARED)) on the same, or another, channel initiator. You can also start it by specifying disposition = A (CHLDISP(SHARED)).

• If the channel is in the starting or retry state, you can restart it by specifying disposition = S (CHLDISP(FIXSHARED)) on the same or a different channel initiator. You can also start it by specifying disposition = A (CHLDISP(SHARED)).

• The channel is eligible to be trigger started when it goes into the inactive state. Shared channels that are trigger started always have a shared disposition (CHLDISP(SHARED)).

• The channel is eligible to be started with CHLDISP(FIXSHARED), on any channel initiator, when it goes into the inactive state. You can also start it by specifying disposition = A (CHLDISP(SHARED)).

• The channel is not recovered by any other active channel initiator in the queue-sharing group when the channel initiator on which it was started is stopped with SHARED(RESTART), or when the channel initiator terminates abnormally. The channel is recovered only when the channel initiator on which

Figure 41. Starting a channel

Select the disposition of the channel instance and on which queue manager it is to be started.

Press Enter to start the channel.

Starting a shared channel
To start a shared channel, and keep it on a nominated channel initiator, use disposition = S (on the START CHANNEL command, specify CHLDISP(FIXSHARED)).

When you start a channel in this way, the following rules apply to that channel:

• You can stop the channel from any queue manager in the queue-sharing group. You can do this even if the channel initiator on which it was started is not running at the time you issue the stop-channel request. When the channel has stopped, you can restart it by specifying disposition = S (CHLDISP(FIXSHARED)) on the same, or another, channel initiator. You can also start it by specifying disposition = A (CHLDISP(SHARED)).

• If the channel is in the starting or retry state, you can restart it by specifying disposition = S (CHLDISP(FIXSHARED)) on the same or a different channel initiator. You can also start it by specifying disposition = A (CHLDISP(SHARED)).

• The channel is eligible to be trigger started when it goes into the inactive state. Shared channels that are trigger started always have a shared disposition (CHLDISP(SHARED)).

• The channel is eligible to be started with CHLDISP(FIXSHARED), on any channel initiator, when it goes into the inactive state. You can also start it by specifying disposition = A (CHLDISP(SHARED)).

• The channel is not recovered by any other active channel initiator in the queue-sharing group when the channel initiator on which it was started is stopped with SHARED(RESTART), or when the channel initiator terminates abnormally. The channel is recovered only when the channel initiator on which
Testing a channel

To test a channel using the MQSC commands, use PING CHANNEL.

Using the operations and control panels, starting from the initial panel, complete these fields and press Enter:

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Action</td>
<td>5 (Perform &quot;&quot;)</td>
</tr>
<tr>
<td>Object type</td>
<td>SENDER, SERVER, or CHANNEL.</td>
</tr>
<tr>
<td>Name</td>
<td>CHANNEL.TO.USE</td>
</tr>
<tr>
<td>Disposition</td>
<td>The disposition of the channel object.</td>
</tr>
</tbody>
</table>

The Perform a Channel Function panel is displayed. The text following the panel explains how to use the panel:

```
Perform a Channel Function
Select function type, complete fields, then press Enter.

Function type .............. 1. Reset 3. Resolve with commit
                            2. Ping    4. Resolve with backout

Channel name .............. : CHANNEL.TO.USE
Channel type .............. : SENDER
Description .............. : Description of CHANNEL.TO.USE

Disposition .............. P = Private on MQ25
                          S = Shared on MQ25
                          A = Shared on any queue manager

Sequence number for reset .. 1 - 999999999
Data length for ping ...... 16 - 32768

Command ===> ________________________________________________________________

F1=Help  F2=Split  F3=Exit  F9=Swap  F10=Messages  F12=Cancel
```

Figure 42. Testing a channel

Select function type 2 (ping).

Select the disposition of the channel for which the test is to be done and on which queue manager it is to be tested.

The data length is initially set to 16. Change it if you want and press Enter.

Resetting message sequence numbers for a channel

To reset channel sequence numbers using the MQSC commands, use RESET CHANNEL.
Resetting message sequence numbers

Using the operations and control panels, starting from the initial panel, complete these fields and press Enter:

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Action</td>
<td>5 (Perform)</td>
</tr>
<tr>
<td>Object type</td>
<td>channel type (for example SENDER) or CHANNEL</td>
</tr>
<tr>
<td>Name</td>
<td>CHANNEL.TO.USE</td>
</tr>
<tr>
<td>Disposition</td>
<td>The disposition of the channel object.</td>
</tr>
</tbody>
</table>

The Perform a Channel Function panel is displayed (see Figure 42 on page 351).

Select Function type 1 (reset).

Select the disposition of the channel for which the reset is to be done and on which queue manager it is to be done.

The sequence number field is initially set to one. Change this if you want, and press Enter.

Resolving in-doubt messages on a channel

To resolve in-doubt messages on a channel using the MQSC commands, use RESOLVE CHANNEL.

Using the operations and control panels, starting from the initial panel, complete these fields and press Enter:

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Action</td>
<td>5 (Perform)</td>
</tr>
<tr>
<td>Object type</td>
<td>SENDER, SERVER, or CHANNEL</td>
</tr>
<tr>
<td>Name</td>
<td>CHANNEL.TO.USE</td>
</tr>
<tr>
<td>Disposition</td>
<td>The disposition of the object.</td>
</tr>
</tbody>
</table>

The Perform a Channel Function panel is displayed (see Figure 42 on page 351).

Select Function type 3 or 4 (resolve with commit or backout). (See “In-doubt channels” on page 70 for more information.)

Select the disposition of the channel for which resolution is to be done and which queue manager it is to be done on. Press Enter.

Stopping a channel

To stop a channel using the MQSC commands, use STOP CHANNEL.

Using the operations and control panels, starting from the initial panel, complete these fields and press Enter:

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Action</td>
<td>7 (Stop)</td>
</tr>
<tr>
<td>Object type</td>
<td>channel type (for example SENDER) or CHANNEL</td>
</tr>
<tr>
<td>Name</td>
<td>CHANNEL.TO.USE</td>
</tr>
<tr>
<td>Disposition</td>
<td>The disposition of the object.</td>
</tr>
</tbody>
</table>
The Stop a Channel panel is displayed. The text following the panel explains how to use the panel:

Stop a Channel

Complete fields, then press Enter to stop channel.

Channel name ..........: CHANNEL.TO.USE
Channel type ..........: SENDER
Description ..........: Description of CHANNEL.TO.USE

Disposition ..........: P  P=Private on MQ25
                      A=Shared on any queue manager

Stop mode ..........: 1  1. Quiesce  2. Force
Stop status ........: 1  1. Stopped  2. Inactive

Queue manager ........: ________________________________________________
Connection name ....: ________________________________________________

Command ===> ________________________________________________________________

F1=Help  F2=Split  F3=Exit  F9=Swap  F10=Messages  F12=Cancel

Figure 43. Stopping a channel

Select the disposition of the channel for which the stop is to be done and on which queue manager it is to be stopped.

Choose the stop mode that you require:

Quiesce

The channel will stop when the current message is completed and the batch will then be ended, even if the batch size value has not been reached and there are messages already waiting on the transmission queue. No new batches will be started. This is the default.

Force

The channel stops immediately. If a batch of messages is in progress, an ‘in-doubt’ situation may result.

Choose the queue manager and connection name for the channel you want to stop.

Choose the status that you require:

Stopped

The channel will not be restarted automatically, and must be restarted manually. This is the default value if no queue manager or connection name is specified. If a name is specified, it is not allowed.

Inactive

The channel will be restarted automatically when required. This is the default value if a queue manager or connection name is specified.

Press Enter to stop the channel.

Usage notes

This section gives some usage notes about stopping a channel:
Stopping a channel

- If a shared channel is in a retry state and the channel initiator on which it was started is not running, a STOP request for the channel is issued on the queue manager where the command was entered.

See “Stopping and quiescing channels” on page 67 for more information. For information about restarting stopped channels, see “Restarting stopped channels” on page 69.

Displaying channel status

To display the status of a channel or a set of channels using the MQSC commands, use DISPLAY CHSTATUS.

Note: Displaying channel status information may take some time if you have lots of channels.

Using the operations and control panels on the List Channel panel (see Figure 38 on page 343), a summary of the channel status is shown for each channel as follows:

- **INACTIVE** No connections are active
- **status** One connection is active
- **nnn status** More than one connection is current and all current connections have the same status
- **nnn CURRENT** More than one connection is current and the current connections do not all have the same status
- **Blank** WebSphere MQ is unable to determine how many connections are active (for example, because the channel initiator is not running)
  Note: For channel objects with the disposition GROUP, no status is displayed.

where **nnn** is the number of active connections, and **status** is one of the following:

- **INIT** INITIALIZING
- **BIND** BINDING
- **START** STARTING
- **RUN** RUNNING
- **STOP** STOPPING or STOPPED
- **RETRY** RETRYING
- **REQST** REQUESTING

To display more information about the channel status, press the Status key (F11) on the List Channel or the Display, or Alter channel panels to display the List Channels - Current Status panel (see Figure 44 on page 355).
The values for status are as follows:

INIT INITIALIZING
BIND BINDING
START STARTING
RUN RUNNING
STOP STOPPING or STOPPED
RETRY RETRYING
REQST REQUESTING
DOUBT STOPPED and INDOUBT(YES)

See "Channel states" on page 61 for more information about these.

You can press F11 to see a similar list of channel connections with saved status; press F11 to get back to the current list. Note that the saved status does not apply until at least one batch of messages has been transmitted on the channel.

Use a slash (/) to select a connection and press Enter. The Display Channel Connection Current Status panels are displayed.

### Displaying cluster channels

To display all the cluster channels that have been defined (explicitly or using auto-definition), use the MQSC command, DISPLAY CLUSQMGR.

Using the operations and control panels, starting from the initial panel, complete these fields and press Enter:

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Action</td>
<td>1 (Display)</td>
</tr>
<tr>
<td>Object type</td>
<td>CLUSCHL</td>
</tr>
<tr>
<td>Name</td>
<td>*</td>
</tr>
</tbody>
</table>
Displaying cluster channels

You are presented with a panel like figure 45, in which the information for each cluster channel occupies three lines, and includes its channel, cluster, and queue manager names. For cluster-sender channels, the overall state is shown.

<table>
<thead>
<tr>
<th>Channel name</th>
<th>Connection name</th>
<th>State</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;&gt; *</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TO.MQ90.T</td>
<td>HURSLEY.MACH90.COM(1590)</td>
<td>N</td>
</tr>
<tr>
<td>- CLUSRCVR</td>
<td>VJH0IT</td>
<td></td>
</tr>
<tr>
<td>- MQ90</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TO.MQ95.T</td>
<td>HURSLEY.MACH95.COM(1595)</td>
<td>RUN</td>
</tr>
<tr>
<td>- CLUSSDRA</td>
<td>VJH0IT</td>
<td>N</td>
</tr>
<tr>
<td>- MQ95</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TO.MQ96.T</td>
<td>HURSLEY.MACH96.COM(1596)</td>
<td>RUN</td>
</tr>
<tr>
<td>- CLUSSDRB</td>
<td>VJH0IT</td>
<td>N</td>
</tr>
<tr>
<td>- MQ96</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

******** End of list ********

Figure 45. Listing cluster channels

To display full information about one or more channels, type Action code 1 against their names and press Enter. Use Action codes 5, 6, or 7 to perform functions (such as ping, resolve, and reset), and start or stop a cluster channel.

To display more information about the channel status, press the Status key (F11).
Chapter 24. Preparing WebSphere MQ for z/OS

This chapter describes the WebSphere MQ for z/OS preparations you need to make before you can start to use distributed queuing. If you are using queue-sharing groups, see Chapter 33, “Preparing WebSphere MQ for z/OS for DQM with queue-sharing groups” on page 439. (If you want to use CICS ISC for distributed queuing, see Chapter 29, “Preparing WebSphere MQ for z/OS when using CICS” on page 413.) Much of the information here applies equally to MQSeries for OS/390 or MVS/ESA.

To enable distributed queuing, you must perform the following three tasks:

1. Customize the distributed queuing facility and define the WebSphere MQ objects required; this is described in the WebSphere MQ for z/OS Concepts and Planning Guide and the WebSphere MQ for z/OS System Setup Guide.
2. Define access security; this is described in the WebSphere MQ for z/OS System Setup Guide.
3. Set up your communications; this is described in Chapter 25, “Setting up communication for z/OS” on page 361.

Defining DQM requirements to WebSphere MQ

In order to define your distributed-queuing requirements, you have to:

- Define the channel initiator procedures and data sets
- Define the channel definitions
- Define the queues and other objects
- Define access security

See the WebSphere MQ for z/OS Concepts and Planning Guide for information about these tasks.

Defining WebSphere MQ objects

Use one of the WebSphere MQ command input methods to define WebSphere MQ objects. Refer to Chapter 23, “Monitoring and controlling channels on z/OS” on page 341 for information about defining objects.

Transmission queues and triggering channels

Define the following:

- A local queue with the usage of XMITQ for each sending message channel.
- Remote queue definitions.

A remote queue object has three distinct uses, depending upon the way the name and content are specified:

- Remote queue definition
- Queue manager alias definition
- Reply-to queue alias definition

This is shown in Table 2 on page 37.

Use the TRIGDATA field on the transmission queue to trigger the specified channel. For example:
Defining WebSphere MQ objects

```
DEFINE QLOCAL(MYXMITQ) USAGE(XMITQ) TRIGGER(YES) +
  INITQ(SYSTEM.CHANNEL.INITQ) TRIGDATA(MYCHANNEL)
DEFINE CHL(MYCHANNEL) CHLTYPE(SDR) TRPTYPE(TCP) +
  XMITQ(MYXMITQ) CONNAME('9.20.9.30(1555)')
```

The supplied sample CSQ4INYD gives additional examples of the necessary definitions.

Synchronization queue

DQM requires a queue for use with sequence numbers and logical units of work identifiers (LUWID). You must ensure that a queue is available with the name SYSTEM.CHANNEL.SYNCQ (see WebSphere MQ for z/OS Concepts and Planning Guide). This queue must be available otherwise the channel initiator cannot start.

Make sure that you define this queue using INDXTYPE(MSGID). This will improve the speed at which they can be accessed.

Channel command queues

You need to ensure that a channel command queue exists for your system with the name SYSTEM.CHANNEL.INITQ.

If the channel initiator detects a problem with the SYSTEM.CHANNEL.INITQ, it will be unable to continue normally until the problem is corrected. The problem could be one of the following:

- The queue is full
- The queue is not enabled for put
- The page set that the queue is on is full
- The channel initiator does not have the correct security authorization to the queue

If the definition of the queue is changed to GET(DISABLED) while the channel initiator is running, it will not be able to get messages from the queue, and will terminate.
**Starting the channel initiator**

Triggering is implemented using the channel initiator. On WebSphere MQ for z/OS, this is started with the MQSC command `START CHINIT PARM(xparm)` where `xparm` is your channel initiator parameter module.

**Stopping the channel initiator**

The channel initiator is stopped automatically when you stop the queue manager. If you need to stop the channel initiator but not the queue manager, you can use the MQSC command `STOP CHINIT`.

---

**Other things to consider**

Here are some other topics that you should consider when preparing WebSphere MQ for distributed queue management.

**Operator Messages**

Because the channel initiator uses a number of asynchronously operating dispatchers, operator messages could appear on the log out of chronological sequence.

**Channel operation commands**

Channel operation commands generally involve two stages. When the command syntax has been checked and the existence of the channel verified, a request is sent to the channel initiator, and message CSQM134I or CSQM137I is sent to the command issuer to indicate the completion of the first stage. When the channel initiator has processed the command, further messages indicating its success or otherwise are sent to the command issuer along with message CSQ9022I or CSQ9023I respectively. Any error messages generated could also be sent to the z/OS console.

All cluster commands except DISPLAY CLUSQMGR, however, work asynchronously. Commands that change object attributes update the object and send a request to the channel initiator, and commands for working with clusters are checked for syntax and a request is sent to the channel initiator. In both cases, message CSQM130I is sent to the command issuer indicating that a request has been sent; this is followed by message CSQ9022I to indicate that the command has completed successfully, in that a request has been sent. It does not indicate that the cluster request has completed successfully. The requests sent to the channel initiator are processed asynchronously, along with cluster requests received from other members of the cluster. In some cases, these requests have to be sent to the whole cluster to determine if they are successful or not. Any errors are reported to the z/OS on the system where the channel initiator is running. They are not sent to the command issuer.

**Undelivered-message queue**

A DLQ handler is provided with WebSphere MQ for z/OS. See [WebSphere MQ for z/OS System Administration Guide](#) for more information.

**Queues in use**

MCAs for receiver channels may keep the destination queues open even when messages are not being transmitted; this results in the queues appearing to be ‘in use’.
Channel operation considerations

**Security changes**

If you change security access for a user ID, the change may not take effect immediately. (See one of [WebSphere MQ for z/OS Concepts and Planning Guide](#), [WebSphere MQ for z/OS System Setup Guide](#) and [WebSphere MQ for z/OS System Administration Guide](#) for more information.)

**Communications stopped**

**TCP**

If TCP is stopped for some reason and then restarted, the WebSphere MQ for z/OS TCP listener waiting on a TCP port is stopped.

If you are not using the OpenEdition® sockets interface, (for example, if you are using the IUCV interface or the Computer Associates SOLVE/TCPaccess interface,) the channel initiator must be stopped and manually restarted when TCP returns. Then, the listener must also be manually restarted to resume communications.

If you are using the OpenEdition sockets interface, automatic channel reconnect allows the channel initiator to detect that TCP/IP is not available and to automatically restart the TCP/IP listener when TCP/IP returns. This alleviates the need for operations staff to notice the problem with TCP/IP and manually restart the listener. While the listener is out of action, the channel initiator can also be used to retry the listener at the interval specified by LSTRTMR in the channel initiator parameter module. These attempts can continue until TCP/IP returns and the listener successfully restarts automatically. For information about LSTRTMR, see the [WebSphere MQ for z/OS System Setup Guide](#).

**LU6.2**

If APPC is stopped, the listener is also stopped. Again, in this case, the listener automatically retries at the LSTRTMR interval so that, if APPC restarts, the listener can restart too.

If the DB2 fails, shared channels that are already running continue to run, but any new channel start requests will fail. When the DB2 is restored new requests are able to complete.

---

**z/OS Automatic Restart Management (ARM)**

Automatic restart management (ARM) is a z/OS recovery function that can improve the availability of specific batch jobs or started tasks (for example, subsystems), and therefore result in a faster resumption of productive work.

To use ARM, you must set up your queue managers and channel initiators in a particular way to make them restart automatically. For information about this, see [WebSphere MQ for z/OS Concepts and Planning Guide](#).
Chapter 25. Setting up communication for z/OS

DQM is a remote queuing facility for WebSphere MQ. It provides channel control programs for the queue manager that form the interface to communication links. These links are controllable by the system operator. The channel definitions held by distributed queuing management use these connections.

When a distributed-queuing management channel is started, it tries to use the connection specified in the channel definition. For this to succeed, it is necessary for the connection to be defined and available. This section explains how to do this.

You might also find it helpful to refer to Chapter 26, “Example configuration - IBM WebSphere MQ for z/OS” on page 365. For z/OS using CICS, see Chapter 30, “Setting up CICS communication for WebSphere MQ for z/OS” on page 417. If you are using queue sharing groups, see Chapter 34, “Setting up communication for WebSphere MQ for z/OS using queue-sharing groups” on page 445.

Deciding on a connection

There are two forms of communication protocol that can be used:
• TCP
• LU 6.2 through APPC/MVS

Each channel definition must specify only one protocol as the transmission protocol (Transport Type) attribute. A queue manager can use more than one protocol to communicate.

Defining a TCP connection

The TCP address space name must be specified in the TCP system parameters data set, tcpip.TCPIP.DATA. In the data set, a “TCPIPJOBNAME TCPIP_proc” statement must be included.

The channel initiator address space must have authority to read the data set. The following techniques can be used to access your TCPIP.DATA data set, depending on which TCP/IP product and interface you are using:
• Environment variable, RESOLVER_CONFIG
• HFS file, /etc/resolv.conf
• //SYSTCPD DD statement
• //SYSTCPDD DD statement
• jobname/userid.TCPIP.DATA
• SYS1.TCPPARMS(TCPDATA)
• zapname.TCPIP.DATA

You must also be careful to specify the high-level qualifier for TCP/IP correctly.

You should have a suitably configured Domain Name System (DNS) server, capable of both Name to IP Address translation and IP Address to Name translation.
Deciding on a connection

For more information, see the following:
- *TCP/IP OpenEdition: Planning and Release Guide, SC31-8303*
- *z/OS Unix System Services Planning, GA22-7800*
- Your SOLVE:TCPaccess documentation

Each TCP channel when started will use TCP resources; you may need to adjust the following parameters in your PROFILE.TCPIP configuration data set:

**ACBPOOLSIZE**
Add one per started TCP channel, plus one

**CCBPOOLSIZE**
Add one per started TCP channel, plus one per DQM dispatcher, plus one

**DATABUFFERPOOLSIZE**
Add two per started TCP channel, plus one

**MAXFILEPROC**
Controls how many channels each dispatcher in the channel initiator can handle.

This parameter is specified in the BPXPRMxx member of SYS1.PARMLIB. If you are using OpenEdition sockets, ensure that you specify a value large enough for your needs.

Sending end

The connection name (CONNAME) field in the channel definition should be set to either the TCP network address of the target, in dotted decimal form (for example 9.20.9.30) or the host name (for example MVSHUR1). If the connection name is a host name, a TCP name server is required to convert the host name into a TCP host address. (This is a function of TCP, not WebSphere MQ.)

On the initiating end of a connection (sender, requester, and server channel types) it is possible to provide an optional port number for the connection, for example:

**Connection name**
9.20.9.30(1555)

This case the initiating end will attempt to connect to a receiving program listening on port 1555.

Receiving on TCP

Receiving channel programs are started in response to a startup request from the sending channel. To do this, a listener program has to be started to detect incoming network requests and start the associated channel. You start this listener program with the START LISTENER command, or using the operations and control panels.

By default, the TCP Listener program uses port 1414 and listens on all addresses available to your TCP stack. You may start your TCP listener program to only listen on a specific address or hostname by specifying IPADDR in the START LISTENER command. (For more information, see Chapter 33, “Preparing WebSphere MQ for z/OS for DQM with queue-sharing groups” on page 439, “Listeners”.)

Using the TCP listener backlog option

When receiving on TCP/IP, a maximum number of outstanding connection requests is set. This can be considered a backlog of requests waiting on the TCP/IP port for the listener to accept the request.
Deciding on a connection

The default listener backlog value on z/OS is 255. If the backlog reaches this value, the TCP/IP connection is rejected and the channel will not be able to start.

For MCA channels, this results in the channel going into a RETRY state and retrying the connection at a later time.

For client connections, the client receives an MQRC_Q_MGR_NOT_AVAILABLE reason code from MQCONN and should retry the connection at a later time.

Defining an LU6.2 connection

APPC/MVS setup

Each instance of the channel initiator must have the name of the LU that it is to use defined to APPC/MVS, in the APPCPMxx member of SYS1.PARMLIB, as in the following example:

\texttt{LUADD ACBNAME(\textit{luname}) NOSCHED TPDATA(CSQ.APPCTP)}

\textit{luname} is the name of the logical unit to be used. \texttt{NOSCHED} is required; \texttt{TPDATA} is not used. No additions are necessary to the ASCHPMxx member, or to the APPC/MVS TP profile data set.

The side information data set must be extended to define the connections used by DQM. See the supplied sample CSQ4SIDE for details of how to do this using the APPC utility program ATBSDFMU. For details of the TPNAME values to use, see the \textit{Multiplatform APPC Configuration Guide} ("Red Book") and the following table for information:

\textit{Table 32. Settings on the local z/OS system for a remote queue manager platform}

<table>
<thead>
<tr>
<th>Remote platform</th>
<th>TPNAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>z/OS, OS/390, or MVS/ESA</td>
<td>The same as TPNAME in the corresponding side information on the remote queue manager.</td>
</tr>
<tr>
<td>z/OS, OS/390, or MVS/ESA using CICS</td>
<td>CKRC (sender) CKSV (requester) CKRC (server)</td>
</tr>
<tr>
<td>OS/400</td>
<td>The same as the compare value in the routing entry on the OS/400 system.</td>
</tr>
<tr>
<td>OS/2</td>
<td>As specified in the OS/2 Run Listener command, or defaulted from the OS/2 queue manager configuration file.</td>
</tr>
<tr>
<td>Compaq OpenVMS Alpha</td>
<td>As specified in the Digital OVMS Run Listener command.</td>
</tr>
<tr>
<td>Compaq NonStop Kernel</td>
<td>The same as the TPNAME specified in the receiver-channel definition.</td>
</tr>
<tr>
<td>Other UNIX systems</td>
<td>The same as TPNAME in the corresponding side information on the remote queue manager.</td>
</tr>
<tr>
<td>Windows</td>
<td>As specified in the Windows Run Listener command, or the invokable Transaction Program that was defined using TpSetup on Windows.</td>
</tr>
</tbody>
</table>

If you have more than one queue manager on the same machine, ensure that the TPnames in the channel definitions are unique.

See the \textit{Multiplatform APPC Configuration Guide} also for information about the VTAM definitions that may be required.
Deciding on a connection

In an environment where the queue manager is communicating via APPC with a queue manager on the same or another z/OS system, ensure that either the VTAM definition for the communicating LU specifies SECACTPT(ALREADYV), or that there is a RACF® APPCLU profile for the connection between LUs, which specifies CONVSEC(ALREADYV).

The z/OS command VARY ACTIVE must be issued against both base and listener LUs before attempting to start either inbound or outbound communications.

Connecting to APPC/MVS (LU 6.2)
The connection name (CONNAME) field in the channel definition should be set to the symbolic destination name, as specified in the side information data set for APPC/MVS.

The LU name to use (defined to APPC/MVS as described above) must also be specified in the channel initiator parameters. It must be set to the same LU that will be used for receiving by the listener.

The channel initiator uses the “SECURITY(SAME)” APPC/MVS option, so it is the user ID of the channel initiator address space that is used for outbound transmissions, and will be presented to the receiver.

Receiving on LU 6.2
Receiving MCAs are started in response to a startup request from the sending channel. To do this, a listener program has to be started to detect incoming network requests and start the associated channel. The listener program is an APPC/MVS server. You start it with the START LISTENER command, or using the operations and control panels. You must specify the LU name to use by means of a symbolic destination name defined in the side information data set. The local LU so identified must be the same as that used for outbound transmissions, as set in the channel initiator parameters.
Chapter 26. Example configuration - IBM WebSphere MQ for z/OS

This chapter gives an example of how to set up communication links from WebSphere MQ for z/OS or MQSeries for OS/390 or MVS/ESA to WebSphere MQ products on the following platforms:

- OS/2
- Windows
- AIX
- Compaq Tru64 UNIX
- HP-UX
- AT&T GIS UNIX\(^7\)
- Solaris
- OS/400
- VSE/ESA

You can also connect any of the following:

- z/OS to z/OS
- z/OS to MVS/ESA
- MVS/ESA to MVS/ESA

First it describes the parameters needed for an LU 6.2 connection; then it describes:

- “Establishing an LU 6.2 connection” on page 368
- “Establishing a TCP connection” on page 371

Once the connection is established, you need to define some channels to complete the configuration. This is described in “WebSphere MQ for z/OS configuration” on page 371.

See Chapter 7, “Example configuration chapters in this book” on page 101 for background information about this chapter and how to use it.

Configuration parameters for an LU 6.2 connection

Table 33 on page 366 presents a worksheet listing all the parameters needed to set up communication from z/OS to one of the other WebSphere MQ platforms. The worksheet shows examples of the parameters, which have been tested in a working environment, and leaves space for you to fill in your own values. An explanation of the parameter names follows the worksheet. Use the worksheet in this chapter in conjunction with the worksheet in the chapter for the platform to which you are connecting.

---

\(^7\) This platform has become NCR UNIX SVR4 MP-RAS, R3.0

© Copyright IBM Corp. 1993, 2002

365
The steps required to set up an LU 6.2 connection are described in "Establishing an LU 6.2 connection" on page 368 with numbered cross references to the parameters on the worksheet.

### Configuration worksheet

Use this worksheet to record the values you use for your configuration. Where numbers appear in the Reference column they indicate that the value must match that in the appropriate worksheet elsewhere in this book. The examples that follow in this chapter refer back to the values in the ID column. The entries in the Parameter Name column are explained in "Explanation of terms" on page 367.

#### Table 33. Configuration worksheet for z/OS using LU 6.2

<table>
<thead>
<tr>
<th>ID</th>
<th>Parameter Name</th>
<th>Reference</th>
<th>Example Used</th>
<th>User Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Command prefix</td>
<td>+cpf</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Network ID</td>
<td>NETID</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Node name</td>
<td>MVSPU</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Local LU name</td>
<td>MVSLU</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Symbolic destination</td>
<td>M1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>Modename</td>
<td>#INTER</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>Local Transaction Program name</td>
<td>MQSERIES</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>LAN destination address</td>
<td>400074511092</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Definition for local node**

<table>
<thead>
<tr>
<th>ID</th>
<th>Parameter Name</th>
<th>Reference</th>
<th>Example Used</th>
<th>User Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>13</td>
<td>Symbolic destination</td>
<td>M2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>Modename</td>
<td>#INTER</td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>Remote Transaction Program name</td>
<td>MQSERIES</td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>Partner LU name</td>
<td>OS2LU</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Connection to an OS/2 system**

The values in this section of the table must match those used in Table 15 on page 144 as indicated.

<table>
<thead>
<tr>
<th>ID</th>
<th>Parameter Name</th>
<th>Reference</th>
<th>Example Used</th>
<th>User Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>13</td>
<td>Symbolic destination</td>
<td>M2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>Modename</td>
<td>#INTER</td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>Remote Transaction Program name</td>
<td>MQSERIES</td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>Partner LU name</td>
<td>OS2LU</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Connection to a Windows system**

The values in this section of the table must match those used in Table 17 on page 168 as indicated.

<table>
<thead>
<tr>
<th>ID</th>
<th>Parameter Name</th>
<th>Reference</th>
<th>Example Used</th>
<th>User Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>13</td>
<td>Symbolic destination</td>
<td>M3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>Modename</td>
<td>#INTER</td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>Remote Transaction Program name</td>
<td>MQSERIES</td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>Partner LU name</td>
<td>WINNTLU</td>
<td></td>
<td></td>
</tr>
<tr>
<td>21</td>
<td>Remote node ID</td>
<td>05D 30F65</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Connection to an AIX system**

The values in this section of the table must match those used in Table 21 on page 197 as indicated.

<table>
<thead>
<tr>
<th>ID</th>
<th>Parameter Name</th>
<th>Reference</th>
<th>Example Used</th>
<th>User Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>13</td>
<td>Symbolic Destination</td>
<td>M4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>Modename</td>
<td>#INTER</td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>Remote Transaction Program name</td>
<td>MQSERIES</td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>Partner LU name</td>
<td>AIXLU</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Connection to an HP-UX system**

The values in this section of the table must match those used in Table 24 on page 219 as indicated.

<table>
<thead>
<tr>
<th>ID</th>
<th>Parameter Name</th>
<th>Reference</th>
<th>Example Used</th>
<th>User Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>13</td>
<td>Symbolic Destination</td>
<td>M5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>Modename</td>
<td>#INTER</td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>Remote Transaction Program name</td>
<td>MQSERIES</td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>Partner LU name</td>
<td>HPUXLU</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Table 33. Configuration worksheet for z/OS using LU 6.2 (continued)

<table>
<thead>
<tr>
<th>ID</th>
<th>Parameter Name</th>
<th>Reference</th>
<th>Example Used</th>
<th>User Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>13</td>
<td>Symbolic Destination</td>
<td></td>
<td>M6</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>Modename</td>
<td></td>
<td>#INTER</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>Remote Transaction Program name</td>
<td>5</td>
<td>MQSERIES</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>Partner LU name</td>
<td></td>
<td>GISLU</td>
<td></td>
</tr>
</tbody>
</table>

Connection to a Solaris system

The values in this section of the table must match those used in Table 28 on page 258 as indicated.

<table>
<thead>
<tr>
<th>ID</th>
<th>Parameter Name</th>
<th>Reference</th>
<th>Example Used</th>
<th>User Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>13</td>
<td>Symbolic destination</td>
<td></td>
<td>M7</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>Modename</td>
<td></td>
<td>#INTER</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>Remote Transaction Program name</td>
<td>8</td>
<td>MQSERIES</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>Partner LU name</td>
<td></td>
<td>SOLARLU</td>
<td></td>
</tr>
</tbody>
</table>

Connection to an iSeries system

The values in this section of the table must match those used in Table 48 on page 521 as indicated.

<table>
<thead>
<tr>
<th>ID</th>
<th>Parameter Name</th>
<th>Reference</th>
<th>Example Used</th>
<th>User Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>13</td>
<td>Symbolic Destination</td>
<td></td>
<td>M8</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>Modename</td>
<td></td>
<td>#INTER</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>Remote Transaction Program name</td>
<td>8</td>
<td>MQSERIES</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>Partner LU name</td>
<td></td>
<td>AS400LU</td>
<td></td>
</tr>
</tbody>
</table>

Connection to a VSE/ESA system

The values in this section of the table must match those used in Table 50 on page 549 as indicated.

<table>
<thead>
<tr>
<th>ID</th>
<th>Parameter Name</th>
<th>Reference</th>
<th>Example Used</th>
<th>User Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>13</td>
<td>Symbolic destination</td>
<td></td>
<td>M9</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>Modename</td>
<td></td>
<td>#INTER</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>Remote Transaction Program name</td>
<td>4</td>
<td>MQ01</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>Partner LU name</td>
<td></td>
<td>VSELU</td>
<td></td>
</tr>
</tbody>
</table>

Explanation of terms

1 **Command prefix**

This is the unique command prefix of your WebSphere MQ for z/OS queue-manager subsystem. The z/OS systems programmer defines this at installation time, in SYS1.PARMLIB(IEFSSNss), and will be able to tell you the value.

2 **Network ID**

The VTAM startup procedure in your installation is partly customized by the ATCSTRxx member of the data set referenced by the DDNAME VTAMLST. The Network ID is the value specified for the NETID parameter in this member. For Network ID you must specify the name of the NETID that owns the WebSphere MQ communications subsystem (WebSphere MQ channel initiator or CICS for z/OS as the case may be). Your network administrator will tell you the value.

3 **Node name**

VTAM, being a low-entry network node, does not have a Control Point name for Advanced Peer-to-Peer Networking (APPN) use. It does however have a system services control point name (SSCPNAME). For node name, you must specify the name of the SSCP that owns the WebSphere MQ communications subsystem (WebSphere MQ channel initiator or CICS for
z/OS and LU 6.2

z/OS as the case may be). This is defined in the same ATCSTRxx member as the Network ID. Your network administrator will tell you the value.

4 **Local LU name**

A logical unit (LU) is software that serves as an interface or translator between a transaction program and the network. It manages the exchange of data between transaction programs. The local LU name is the unique VTAM APPLID of this WebSphere MQ subsystem. Your network administrator will tell you this value.

5 **Symbolic destination**

This is the name you give to the CPI-C side information profile. You need a side information entry for each LU 6.2 listener.

6 **Modename**

This is the name given to the set of parameters that control the LU 6.2 conversation. An entry with this name and similar attributes must be defined at each end of the session. In VTAM, this corresponds to a mode table entry. Your network administrator will assign this to you.

7 **Transaction Program name**

WebSphere MQ applications trying to converse with this queue manager will specify a symbolic name for the program to be run at the receiving end. This will have been specified in the TPNAME attribute on the channel definition at the sender. For simplicity, wherever possible use a transaction program name of MQSERIES, or in the case of a connection to VSE/ESA, where the length is limited to 4 bytes, use MQTP.

See Table 32 on page 363 for more information. If the receiving end is z/OS using CICS, special values are required.

8 **LAN destination address**

This is the LAN destination address that your partner nodes will use to communicate with this host. When you are using a 3745 network controller, it will be the value specified in the LOCADD parameter for the line definition to which your partner is physically connected. If your partner nodes use other devices such as 317X or 6611 devices, the address will have been set during the customization of those devices. Your network administrator will tell you this value.

16 **Partner LU name**

This is the LU name of the WebSphere MQ queue manager on the system with which you are setting up communication. This value is specified in the side information entry for the remote partner.

21 **Remote node ID**

For a connection to Windows, this is the ID of the local node on the Windows system with which you are setting up communication.

---

**Establishing an LU 6.2 connection**

To establish and LU 6.2 connection, there are two steps:

1. Define yourself to the network.
2. Define a connection to the partner.

**Defining yourself to the network**

1. SYSL.PARMLIB(APPCPMxx) contains the startup parameters for APPC. You must add a line to this file to define the local LU name you intend to use for the WebSphere MQ LU 6.2 listener. The line you add should take the form
Specify values for `ACBNAME` and `TPDATA`.

The `NOSCHED` parameter tells APPC that our new LU will not be using the LU 6.2 scheduler (ASCH), but has one of its own. TPDAT refers to the Transaction Program data set in which LU 6.2 stores information about transaction programs. Again, WebSphere MQ will not use this, but it is required by the syntax of the LUADD command.

2. Start the APPC subsystem with the command:

   ```
   START APPC,SUB=MSTR,APPC=xx
   ```

   where `xx` is the suffix of the PARMLIB member in which you added the LU in step 1.

   **Note:** If APPC is already running, it can be refreshed with the command:

   ```
   SET APPC=xx
   ```

   The effect of this is cumulative, that is, APPC will not lose its knowledge of objects already defined to it in this or another PARMLIB member.

3. Add the new LU to a suitable VTAM major node definition. These are typically in `SYS1.VTAMLST`. The APPL definition will look similar to the sample shown in Figure 46.

4. Activate the major node. This can be done with the command:

   ```
   V,NET,ACT,majornode
   ```

5. Add an entry defining your LU to the CPI-C side information data set. Use the APPC utility program ATBSDFMU to do this. Sample JCL is in `thlqual.SCSQPROC(CSQ4SIDE)` (where `thlqual` is the target library high-level qualifier for WebSphere MQ data sets in your installation.)

   The entry you add will look like this:

   ```
   SIADD
   DESTNAME(M1)
   MODENAME(#INTER)
   TPNAME(MQSERIES)
   PARTNER_LU(MVSLU)
   ```

**Figure 46. Channel Initiator APPL definition**
6. Create the channel-initiator parameter module for your queue manager. Sample JCL to do this is in `thlqual.SCSQPROC(CSQ4XPRM)`. You must specify the local LU (4) assigned to your queue manager in the `LUNAME=` parameter of the `CSQ6CHIP` macro.

```csh
//SYSIN DD *
CSQ6CHIP ADAPS=8, X
   ACTCHL=200, X
   ADOPTMCA=NO, X
   ADOPTCHK=ALL, X
   CURRCHL=200, X
   DISPS=5, X
   DNGROUP=, X
   DNSWLM=NO, X
   LSTRTM=60, X
   LU62ARM=, X
   LUNAME=MVSLU, X
   LU62ARM=, X
   LU62CHL=200, X
   OPORTMIN=0, X
   OPORTMAX=0, X
   TCPCHL=200, X
   TCPKEEP=NO, X
   TCPPNAME=TCPIP, X
   TCPNAME=0ESOCKET, X
   TRAXSTR=YES, X
   TRAXTBL=2, X
   SERVICE=0
END
/*

Figure 47. Channel Initiator initialization parameters

7. Modify the job to assemble and link-edit the tailored version of the initiator macro to produce a new load module.
8. Submit the job and verify that it completes successfully.
9. Put the new initialization-parameters module in an APF-authorized user library. Include this library in the STEPLIB concatenation for the channel initiator’s started-task procedure, ensuring that it precedes the library `thlqual.SCSQAUTH`.

Defining a connection to a partner

Note: This example is for a connection to an OS/2 system but the task is the same for other platforms.

Add an entry to the CPI-C side information data set to define the connection. Sample JCL to do this is in `thlqual.SCSQPROC(CSQ4SIDE)`.  

The entry you add will look like this:
```
SIADD
   DESTNAME(M2) 13
   MODENAME(#INTER) 14
   TPNAME(MQSERIES) 15
   PARTNER_LU(OS2LU) 16
```
Establishing a TCP connection

Edit the channel initiator initialization parameters. Sample JCL to do this is in `thqual.SCSQPROC(CSQ4XPRM)`. You must add the name of the TCP address space to the TCPNAME= parameter.

```
//SYSIN DD *
CSQ6CHIP ADAPS=8, X
ACTCHL=200, X
ADOPTMCA=NO, X
ADOPTCHK=ALL, X
CURRCHL=200, X
DISPS=5, X
DNSGROUP=, X
DNSWLM=NO, X
LSTRTRMR=60, X
LUGROUP=, X
LUNAME=MVSLU, X
LU62ARM=, X
LU62CHL=200, X
OPORTMIN=0, X
OPORTMAX=0, X
TCPCHL=200, X
TCPKEEP=NO, X
TCPNAME=TCPIP, X
TCPNAME=TCPIP, X
TCPNAME=TCPIP, X
TCPNAME=OESOCKET, X
TRAXSTR=YES, X
TRAXTBL=2, X
SERVICE=0
END
/*

Figure 48. Channel Initiator initialization parameters

What next?

The TCP connection is now established. You are ready to complete the configuration. Go to “WebSphere MQ for z/OS configuration”.

WebSphere MQ for z/OS configuration

1. Start the channel initiator using the command:
   
   ```
   +cpf START CHINIT PARM(xparms)
   ```

   where `xparms` is the name of the channel-initiator parameter module that you created.

2. Start an LU 6.2 listener using the command:
   
   ```
   +cpf START LSTR LUNAME(M1) TRPTYPE(LU62)
   ```

   The LUNAME of M1 refers to the symbolic name you gave your LU (E). You must specify TRPTYPE(LU62), otherwise the listener will assume you want TCP.

3. Start a TCP listener using the command:
   
   ```
   +cpf START LSTR PORT(2)
   ```

   If you wish to use a port other than 1414 (the default WebSphere MQ port), use the command:
   
   ```
   +cpf START LSTR PORT(1555)
   ```
z/OS configuration

WebSphere MQ channels will not initialize successfully if the channel negotiation detects that the message sequence number is different at each end. You may need to reset this manually.

Channel configuration

The following sections detail the configuration to be performed on the z/OS queue manager to implement the channel described in Figure 32 on page 101.

Examples are given for connecting WebSphere MQ for z/OS and MQSeries for OS/2 Warp. If you wish to connect to another WebSphere MQ product use the appropriate set of values from the table in place of those for OS/2.

Note: The words in bold are user-specified and reflect the names of WebSphere MQ objects used throughout these examples. If you change the names used here, ensure that you also change the other references made to these objects throughout this book. All others are keywords and should be entered as shown.

Table 34. Configuration worksheet for WebSphere MQ for z/OS

<table>
<thead>
<tr>
<th>ID</th>
<th>Parameter Name</th>
<th>Reference</th>
<th>Example Used</th>
<th>User Value</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Definition for local node</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>Queue Manager Name</td>
<td></td>
<td>MVS</td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>Local queue name</td>
<td></td>
<td>MVS.LOCALQ</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Connection to MQSeries for OS/2 Warp</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>Remote queue manager name</td>
<td>A</td>
<td>OS2</td>
<td></td>
</tr>
<tr>
<td>D</td>
<td>Remote queue name</td>
<td></td>
<td>OS2.REMOTEQ</td>
<td></td>
</tr>
<tr>
<td>E</td>
<td>Queue name at remote system</td>
<td>B</td>
<td>OS2.LOCALQ</td>
<td></td>
</tr>
<tr>
<td>F</td>
<td>Transmission queue name</td>
<td></td>
<td>OS2</td>
<td></td>
</tr>
<tr>
<td>G</td>
<td>Sender (LU 6.2) channel name</td>
<td></td>
<td>MVS.OS2.SNA</td>
<td></td>
</tr>
<tr>
<td>H</td>
<td>Sender (TCP) channel name</td>
<td></td>
<td>MVS.OS2.TCP</td>
<td></td>
</tr>
<tr>
<td>I</td>
<td>Receiver (LU 6.2) channel name</td>
<td></td>
<td>OS2.MVS.SNA</td>
<td></td>
</tr>
<tr>
<td>J</td>
<td>Receiver (TCP) channel name</td>
<td></td>
<td>OS2.MVS.TCP</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Connection to WebSphere MQ for Windows</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>Remote queue manager name</td>
<td>A</td>
<td>WINNT</td>
<td></td>
</tr>
<tr>
<td>D</td>
<td>Remote queue name</td>
<td></td>
<td>WINNT.REMOTEQ</td>
<td></td>
</tr>
<tr>
<td>E</td>
<td>Queue name at remote system</td>
<td>B</td>
<td>WINNT.LOCALQ</td>
<td></td>
</tr>
<tr>
<td>F</td>
<td>Transmission queue name</td>
<td></td>
<td>WINNT</td>
<td></td>
</tr>
<tr>
<td>G</td>
<td>Sender (LU 6.2) channel name</td>
<td></td>
<td>MVS.WINNT.SNA</td>
<td></td>
</tr>
<tr>
<td>H</td>
<td>Sender (TCP) channel name</td>
<td></td>
<td>MVS.WINNT.TCP</td>
<td></td>
</tr>
<tr>
<td>I</td>
<td>Receiver (LU 6.2) channel name</td>
<td></td>
<td>WINNT.MVS.SNA</td>
<td></td>
</tr>
<tr>
<td>J</td>
<td>Receiver (TCP/IP) channel name</td>
<td></td>
<td>WINNT.MVS.TCP</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Connection to WebSphere MQ for AIX</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>Remote queue manager name</td>
<td>A</td>
<td>AIX</td>
<td></td>
</tr>
<tr>
<td>D</td>
<td>Remote queue name</td>
<td></td>
<td>AIX.REMOTEQ</td>
<td></td>
</tr>
<tr>
<td>E</td>
<td>Queue name at remote system</td>
<td>B</td>
<td>AIX.LOCALQ</td>
<td></td>
</tr>
<tr>
<td>F</td>
<td>Transmission queue name</td>
<td></td>
<td>AIX</td>
<td></td>
</tr>
<tr>
<td>ID</td>
<td>Parameter Name</td>
<td>Reference</td>
<td>Example Used</td>
<td>User Value</td>
</tr>
<tr>
<td>----</td>
<td>-------------------------------------</td>
<td>------------------</td>
<td>------------------</td>
<td>------------------</td>
</tr>
<tr>
<td>G</td>
<td>Sender (LU 6.2) channel name</td>
<td></td>
<td>MVS.AIX.SNA</td>
<td></td>
</tr>
<tr>
<td>H</td>
<td>Sender (TCP/IP) channel name</td>
<td></td>
<td>MVS.AIX.TCP</td>
<td></td>
</tr>
<tr>
<td>I</td>
<td>Receiver (LU 6.2) channel name</td>
<td>G</td>
<td>AIX.MVS.SNA</td>
<td></td>
</tr>
<tr>
<td>J</td>
<td>Receiver (TCP/IP) channel name</td>
<td>F</td>
<td>AIX.MVS.TCP</td>
<td></td>
</tr>
</tbody>
</table>

**Connection to MQSeries for Compaq Tru64 UNIX**

The values in this section of the table must match those used in Table 23 on page 216 as indicated.

| C  | Remote queue manager name          |                  | DECUX            |                  |
| D  | Remote queue name                  |                  | DECUX.REMOTEQ    |                  |
| E  | Queue name at remote system        | E                | DECUX.LOCALQ     |                  |
| F  | Transmission queue name            |                  | DECUX            |                  |
| G  | Sender (LU 6.2) channel name        |                  | DECUX.MVS.TCP    |                  |
| J  | Receiver (TCP) channel name        | F                | MVS.DECUX.TCP    |                  |

**Connection to WebSphere MQ for HP-UX**

The values in this section of the table must match those used in Table 25 on page 239 as indicated.

| C  | Remote queue manager name          |                  | HPUX             |                  |
| D  | Remote queue name                  |                  | HPUX.REMOTEQ     |                  |
| E  | Queue name at remote system        | E                | HPUX.LOCALQ      |                  |
| F  | Transmission queue name            |                  | HPUX             |                  |
| G  | Sender (LU 6.2) channel name        |                  | MVS.HPUX.SNA     |                  |
| H  | Sender (TCP) channel name           |                  | MVS.HPUX.TCP     |                  |
| I  | Receiver (LU 6.2) channel name      | G                | HPUX.MVS.SNA     |                  |
| J  | Receiver (TCP) channel name         | H                | HPUX.MVS.TCP     |                  |

**Connection to MQSeries for AT&T GIS UNIX**

The values in this section of the table must match those used in Table 27 on page 253 as indicated.

| C  | Remote queue manager name          |                  | GIS              |                  |
| D  | Remote queue name                  |                  | GIS.REMOTEQ      |                  |
| E  | Queue name at remote system        | E                | GIS.LOCALQ       |                  |
| F  | Transmission queue name            |                  | GIS              |                  |
| G  | Sender (LU 6.2) channel name        |                  | MVS.GIS.SNA      |                  |
| H  | Sender (TCP) channel name           |                  | MVS.GIS.TCP      |                  |
| I  | Receiver (LU 6.2) channel name      | G                | GVIS.MVS.SNA     |                  |
| J  | Receiver (TCP) channel name         | H                | GVIS.MVS.TCP     |                  |

**Connection to WebSphere MQ for Solaris**

The values in this section of the table must match those used in Table 30 on page 290 as indicated.

| C  | Remote queue manager name          |                  | SOLARIS          |                  |
| D  | Remote queue name                  |                  | SOLARIS.REMOTEQ  |                  |
| E  | Queue name at remote system        | E                | SOLARIS.LOCALQ   |                  |
| F  | Transmission queue name            |                  | SOLARIS          |                  |
| G  | Sender (LU 6.2) channel name        |                  | MVS.SOLARIS.SNA  |                  |
| H  | Sender (TCP) channel name           |                  | MVS.SOLARIS.TCP  |                  |
| I  | Receiver (LU 6.2) channel name      | G                | SOLARIS.MVS.SNA  |                  |
| J  | Receiver (TCP/IP) channel name      | H                | SOLARIS.MVS.TCP  |                  |
### Table 34. Configuration worksheet for WebSphere MQ for z/OS (continued)

<table>
<thead>
<tr>
<th>ID</th>
<th>Parameter Name</th>
<th>Reference</th>
<th>Example Used</th>
<th>User Value</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><strong>Connection to WebSphere MQ for iSeries</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>The values in this section of the table must match those used in <a href="#">Table 49 on page 535</a> as indicated.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>Remote queue manager name</td>
<td>AS400</td>
<td></td>
<td></td>
</tr>
<tr>
<td>D</td>
<td>Remote queue name</td>
<td>AS400.REMOTEQ</td>
<td></td>
<td></td>
</tr>
<tr>
<td>E</td>
<td>Queue name at remote system</td>
<td>AS400.LOCALQ</td>
<td></td>
<td></td>
</tr>
<tr>
<td>F</td>
<td>Transmission queue name</td>
<td>AS400</td>
<td></td>
<td></td>
</tr>
<tr>
<td>G</td>
<td>Sender (LU 6.2) channel name</td>
<td>MVS.AS400.SNA</td>
<td></td>
<td></td>
</tr>
<tr>
<td>H</td>
<td>Sender (TCP/IP) channel name</td>
<td>MVS.AS400.TCP</td>
<td></td>
<td></td>
</tr>
<tr>
<td>I</td>
<td>Receiver (LU 6.2) channel name</td>
<td>AS400.MVS.SNA</td>
<td></td>
<td></td>
</tr>
<tr>
<td>J</td>
<td>Receiver (TCP/IP) channel name</td>
<td>AS400.MVS.TCP</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Connection to MQSeries for VSE/ESA</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>The values in this section of the table must match those used in <a href="#">Table 51 on page 554</a> as indicated.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>Remote queue manager name</td>
<td>VSE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>D</td>
<td>Remote queue name</td>
<td>VSE.REMOTEQ</td>
<td></td>
<td></td>
</tr>
<tr>
<td>E</td>
<td>Queue name at remote system</td>
<td>VSE.LOCALQ</td>
<td></td>
<td></td>
</tr>
<tr>
<td>F</td>
<td>Transmission queue name</td>
<td>VSE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>G</td>
<td>Sender channel name</td>
<td>MVS.VSE.SNA</td>
<td></td>
<td></td>
</tr>
<tr>
<td>I</td>
<td>Receiver channel name</td>
<td>VSE.MVS.SNA</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### WebSphere MQ for z/OS sender-channel definitions using LU 6.2

**Local Queue**
- Object type: QLOCAL
- Name: OS2
- Usage: X (XmitQ)

**Remote Queue**
- Object type: QREMOTE
- Name: OS2.REMOTEQ
- Name on remote system: OS2.LOCALQ
- Remote system name: OS2
- Transmission queue: OS2

**Sender Channel**
- Channel name: MVS(OS2).SNA
- Transport type: L (LU6.2)
- Transmission queue name: OS2
- Connection name: M2

**WebSphere MQ for z/OS receiver-channel definitions using LU 6.2**

**Local Queue**
- Object type: QLOCAL
- Name: MVS.LOCALQ
- Usage: N (Normal)

**Receiver Channel**
- Channel name: OS2.MVS.SNA

**WebSphere MQ for z/OS sender-channel definitions using TCP**

**Local Queue**
- Object type: QLOCAL
- Name: OS2
- Usage: X (XmitQ)
Remote Queue
Object type : QREMOTE
Name : OS2.REMOTEQ
Name on remote system : OS2.LOCALQ
Remote system name : OS2
Transmission queue : OS2

Sender Channel
Channel name : MVS.OS2.TCP
Transport type : T (TCP)
Transmission queue name : OS2
Connection name : os2.tcipp.hostname

WebSphere MQ for z/OS receiver-channel definitions using TCP

Local Queue
Object type : QLOCAL
Name : MVS.LOCALQ
Usage : N (Normal)

Receiver Channel
Channel name : OS2.MVS.TCP
z/OS configuration
Chapter 27. Message channel planning example for z/OS

This chapter provides a detailed example of how to connect z/OS, OS/390, or MVS/ESA queue managers together so that messages can be sent between them.

The example illustrates the preparations needed to allow an application using queue manager QM1 to put messages on a queue at queue manager QM2. An application running on QM2 can retrieve these messages, and send responses to a reply queue on QM1.

The example illustrates the use of both TCP/IP and LU 6.2 connections. The example assumes that channels are to be triggered to start when the first message arrives on the transmission queue they are servicing.

What the example shows

This example shows the WebSphere MQ commands (MQSC) that you can use in WebSphere MQ for z/OS for DQM.

It involves a payroll query application connected to queue manager QM1 that sends payroll query messages to a payroll processing application running on queue manager QM2. The payroll query application needs the replies to its queries sent back to QM1. The payroll query messages are sent from QM1 to QM2 on a sender-receiver channel called QM1.TO.QM2, and the reply messages are sent back from QM2 to QM1 on another sender-receiver channel called QM2.TO.QM1. Both of these channels are triggered to start as soon as they have a message to send to the other queue manager.

The payroll query application puts a query message to the remote queue “PAYROLL.QUERY” defined on QM1. This remote queue definition resolves to the local queue “PAYROLL” on QM2. In addition, the payroll query application specifies that the reply to the query is sent to the local queue “PAYROLL.REPLY” on QM1. The payroll processing application gets messages from the local queue “PAYROLL” on QM2, and sends the replies to wherever they are required; in this case, local queue “PAYROLL.REPLY” on QM1.

Both queue managers are assumed to be running on z/OS. In the example definitions for TCP/IP, QM1 has a host address of 9.20.9.31 and is listening on port
1411, and QM2 has a host address of 9.20.9.32 and is listening on port 1412. In the definitions for LU 6.2, QM1 is listening on a symbolic luname called LUNAME1 and QM2 is listening on a symbolic luname called LUNAME2. The example assumes that these are already defined on your z/OS system and available for use. To define them, see Chapter 26, “Example configuration - IBM WebSphere MQ for z/OS” on page 365.

The object definitions that need to be created on QM1 are:
- Remote queue definition, PAYROLL.QUERY
- Transmission queue definition, QM2 (default=remote queue manager name)
- Sender channel definition, QM1.TO.QM2
- Receiver channel definition, QM2.TO.QM1
- Reply-to queue definition, PAYROLL.REPLY

The object definitions that need to be created on QM2 are:
- Local queue definition, PAYROLL
- Transmission queue definition, QM1 (default=remote queue manager name)
- Sender channel definition, QM2.TO.QM1
- Receiver channel definition, QM1.TO.QM2

The example assumes that all the SYSTEM.COMMAND.* and SYSTEM.CHANNEL.* queues required to run DQM have been defined as shown in the supplied sample definitions, CSQ4INSG and CSQ4INSX.

The connection details are supplied in the CONNAME attribute of the sender channel definitions.

You can see a diagram of the arrangement in Figure 49 on page 377.

Queue manager QM1 example
The following object definitions allow applications connected to queue manager QM1 to send request messages to a queue called PAYROLL on QM2, and to receive replies on a queue called PAYROLL.REPLY on QM1.

All the object definitions have been provided with the DESCR and REPLACE attributes. The other attributes supplied are the minimum required to make the example work. The attributes that are not supplied take the default values for queue manager QM1.

Run the following commands on queue manager QM1.

Remote queue definition
DEFINE QREMOTE(PAYROLL.QUERY) DESCR('Remote queue for QM2') REPLACE +
PUT(ENABLED) XMITQ(QM2) RNAME(PAYROLL) RQMNAME(QM2)

Note: The remote queue definition is not a physical queue, but a means of directing messages to the transmission queue, QM2, so that they can be sent to queue manager QM2.

Transmission queue definition
DEFINE QLOCAL(QM2) DESCR('Transmission queue to QM2') REPLACE +
USAGE(XMITQ) PUT(ENABLED) GET(ENABLED) TRIGGER TRIGTYPE(FIRST) +
TRIGDATA(QM1.TO.QM2) INITQ(SYSTEM.CHANNEL.INITQ)

When the first message is put on this transmission queue, a trigger message is sent to the initiation queue, SYSTEM.CHANNEL.INITQ. The channel initiator gets the
Sender channel definition
For a TCP/IP connection:
DEFINE CHANNEL(QM1.TO.QM2) CHLTYPE(SDR) TRPTYPE(TCP) +
REPLACE DESCR('Sender channel to QM2') XMITQ(QM2) +
CONNAME('9.20.9.32(1412)')

For an LU 6.2 connection:
DEFINE CHANNEL(QM1.TO.QM2) CHLTYPE(SDR) TRPTYPE(LU62) +
REPLACE DESCR('Sender channel to QM2') XMITQ(QM2) +
CONNAME('LUNAME2')

Receiver channel definition
For a TCP/IP connection:
DEFINE CHANNEL(QM2.TO.QM1) CHLTYPE(RCVR) TRPTYPE(TCP) +
REPLACE DESCR('Receiver channel from QM2')

For an LU 6.2 connection:
DEFINE CHANNEL(QM2.TO.QM1) CHLTYPE(RCVR) TRPTYPE(LU62) +
REPLACE DESCR('Receiver channel from QM2')

Reply-to queue definition
DEFINE QLOCAL(PAYROLL.REPLY) REPLACE PUT(ENABLED) GET(ENABLED) +
DESCR('Reply queue for replies to query messages sent to QM2')

The reply-to queue is defined as PUT(ENABLED). This ensures that reply messages can be put to the queue. If the replies cannot be put to the reply-to queue, they are sent to the dead-letter queue on QM1 or, if this queue is not available, remain on transmission queue QM1 on queue manager QM2. The queue has been defined as GET(ENABLED) to allow the reply messages to be retrieved.

Queue manager QM2 example
The following object definitions allow applications connected to queue manager QM2 to retrieve request messages from a local queue called PAYROLL, and to put replies to these request messages to a queue called PAYROLL.REPLY on queue manager QM1.

You do not need to provide a remote queue definition to enable the replies to be returned to QM1. The message descriptor of the message retrieved from local queue PAYROLL contains both the reply-to queue and the reply-to queue manager names. Therefore, as long as QM2 can resolve the reply-to queue manager name to that of a transmission queue on queue manager QM2, the reply message can be sent. In this example, the reply-to queue manager name is QM1 and so queue manager QM2 simply requires a transmission queue of the same name.

All the object definitions have been provided with the DESCR and REPLACE attributes and are the minimum required to make the example work. The attributes that are not supplied take the default values for queue manager QM2.

Run the following commands on queue manager QM2.
Planning examples for z/OS

Local queue definition
DEFINE QLOCAL(PAYROLL) REPLACE PUT(ENABLED) GET(ENABLED) +
DESCR('Local queue for QM1 payroll details')

This queue is defined as PUT(ENABLED) and GET(ENABLED) for the same reason as the reply-to queue definition on queue manager QM1.

Transmission queue definition
DEFINE QLOCAL(QM1) DESCR('Transmission queue to QM1') REPLACE +
USAGE(XMITQ) PUT(ENABLED) GET(ENABLED) TRIGGER TRIGTYPE(FIRST) +
TRIGDATA(QM2.TO.QM1) INITQ(SYSTEM.CHANNEL.INITQ)

When the first message is put on this transmission queue, a trigger message is sent to the initiation queue, SYSTEM.CHANNEL.INITQ. The channel initiator gets the message from the initiation queue and starts the channel identified in the trigger data. The channel initiator can only get trigger messages from SYSTEM.CHANNEL.INITQ so you should not use any other queue as the initiation queue.

Sender channel definition
For a TCP/IP connection:
DEFINE CHANNEL(QM2.TO.QM1) CHLTYPE(SDR) TRPTYPE(TCP) +
REPLACE DESCR('Sender channel to QM1') XMITQ(QM1) +
CONNAME('9.20.9.31(1411)')

For an LU 6.2 connection:
DEFINE CHANNEL(QM2.TO.QM1) CHLTYPE(SDR) TRPTYPE(LU62) +
REPLACE DESCR('Sender channel to QM1') XMITQ(QM1) +
CONNAME('LUNAME1')

Receiver channel definition
For a TCP/IP connection:
DEFINE CHANNEL(QM1.TO.QM2) CHLTYPE(RCVR) TRPTYPE(TCP) +
REPLACE DESCR('Receiver channel from QM1')

For an LU 6.2 connection:
DEFINE CHANNEL(QM1.TO.QM2) CHLTYPE(RCVR) TRPTYPE(LU62) +
REPLACE DESCR('Receiver channel from QM1')

Running the example
When you have created the required objects, you must:
• Start the channel initiator for both queue managers
• Start the listener for both queue managers

The applications can then send messages to each other. Because the channels are triggered to start by the arrival of the first message on each transmission queue, you do not need to issue the START CHANNEL MQSC command.

For details about starting a channel initiator see "Starting a channel initiator" on page 346, and for details about starting a listener see "Starting a channel listener" on page 348.

Expanding this example
This example can be expanded by:
Planning examples for z/OS

- Adding more queue, and channel definitions to allow other applications to send messages between the two queue managers.
- Adding user exit programs on the channels to allow for link encryption, security checking, or additional message processing.
- Using queue manager aliases and reply-to queue aliases to understand more about how these can be used in the organization of your queue manager network.
Planning examples for z/OS
Chapter 28. Monitoring and controlling channels in z/OS with CICS

You monitor and control the channels to remote queue managers from the distributed queue management (DQM) panels. Each z/OS queue manager has a set of DQM CICS transactions for controlling interconnections to compatible remote queue managers using CICS intersystem communication (ISC) facilities.

**Important notice**

Distributed queuing using CICS ISC is retained for compatibility with previous releases; there will be no further enhancements to this function. Therefore you are recommended to use the channel initiator for distributed queuing.

The DQM channel control function

The channel control function provides the administration and control of message channels using CICS between WebSphere MQ for z/OS and compatible systems. See Figure 28 on page 58 for a conceptual picture.

The channel control function consists of CICS panels and programs, a sequence number queue, a channel command queue, and a VSAM file for the channel definitions. The following is a brief description of the components of the channel control function.

- The channel definition file (CDF):
  - Is a VSAM file
  - Is indexed on channel name
  - Holds channel definitions
  - Must be available to the CICS regions in which the channel control program runs, and where the message channel agent (MCA) programs run

- You use channel definition panels to:
  - Create, copy, display, alter, find, and delete channel definitions
  - Start channels, reset channel sequence numbers, stop channels, ping channels, resync channels, and resolve in-doubt messages when links cannot be re-established
  - Display status information about channels

The panels are CICS basic-mapping support maps.

- Sequence numbers and logical unit of work IDs (LUWIDs) are stored in the sequence number queue, SYSTEM.CHANNEL.SEQNO, and are used for channel re-synchronization purposes.
- A channel command queue, SYSTEM.CHANNEL.COMMAND, is used to hold certain commands for channels.
- The programs are a series of CICS transactions, which include transactions for the MCAs. There are different MCAs available for each type of channel. The names are contained in the following table. Other transactions provide channel control, command handling, and trigger monitoring.
Table 35. Program and transaction names

<table>
<thead>
<tr>
<th>Program name</th>
<th>Channel type</th>
<th>CICS transaction ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>CSQKMSGS</td>
<td>Sender</td>
<td>CKSG</td>
</tr>
<tr>
<td>CSQKMSGR</td>
<td>Receiver</td>
<td>CKRC</td>
</tr>
<tr>
<td>CSQKMSGQ</td>
<td>Requester</td>
<td>CKRQ</td>
</tr>
<tr>
<td>CSQKMSGV</td>
<td>Server</td>
<td>CKSV</td>
</tr>
</tbody>
</table>

- A transient data queue CKMQ for error messages.

CICS regions

Figure 50 shows a configuration of two CICS regions, connected to a single queue manager. The regions have multiregion operation (MRO) links to one another, for function shipping of EXEC CICS START commands from the channel control program.

Starting DQM panels

You invoke DQM panels with the CKMC CICS transaction. On invocation, DQM presents you with the main Message Channel List panel. All activity with the other panels follows from selections made on this panel.
The Message Channel List panel

The main panel is called the Message Channel List panel; for an example of it, see [Figure 51]. It has a menu bar with choices you can pull down to reveal the various options you can select for these choices. The work area of the panel is used to present a selection column, and three other columns showing the:

- Full name of each channel
- Type of channel
- CICS system identifier

Keyboard functions

The following sections describe the function, Enter, and Clear keys, as well as what happens if you press any unassigned keys associated with this panel.

Function keys

The function keys control the use of the panel. They are listed below, together with their purpose.

F1   Call help panels
F3   Exit from the panel and the program
F5   Refresh the screen fields with current data
F6   Find a particular channel name
F7   Scroll the panel backward to display more channels
F8   Scroll the panel forward to display more channels
F10  Move the cursor to the menu bar
F12  Cancel pull-down menus or secondary windows, if any, otherwise as F3

Note: Function keys 13 to 24 have the same functions as functions keys 1 to 12, respectively.
Enter key
Pressing the Enter key while the cursor is on a menu-bar choice results in the pull-down menu for that choice appearing.

Pressing the Enter key while the cursor is not on a menu-bar choice and a channel selection has been made selects the default option, Display Settings.

Pressing the Enter key while the cursor is not on a menu-bar choice and no channel selection has been made results in the panel being redisplayed.

Clear key
If you find while typing that what you have typed is not correct, press the Clear key on your terminal to revert all the input fields to their previous state.

For individual fields, use the ‘Erase EOF’, or ‘Ctrl Delete’, depending upon the type of terminal you are using.

Unassigned keys and unavailable choices
If you press a function key, or an attention key that has not been assigned an action, a warning message is displayed that states that the key is invalid.

Selecting a channel
To select a channel, begin at the Message Channel List panel:
1. Move the cursor to the left of the required channel name.
2. Type a slash (/) character.
3. Press F10 to move the cursor to the menu bar, or press the Enter key to browse the channel settings.

If you try to select more than one channel, only the first one you select is valid.

Working with channels
When a channel has been selected, function key F10 moves the cursor to the menu bar (see Table 36). The menu-bar choices are:

Table 36. Message Channel List menu-bar choices

<table>
<thead>
<tr>
<th>Selected</th>
<th>Edit</th>
<th>View</th>
<th>Help</th>
</tr>
</thead>
</table>

Selecting each of these choices causes its pull-down menu to be displayed (see Figure 52 on page 387).

When you select an option that requires further information, such as a channel name, an action window appears with an entry field for the data.

In general, any incorrect input from the keyboard results in a warning message being issued.
Creating a channel

To create a new channel, begin at the Message Channel List panel:

1. Press function key F10 and move the cursor to the Edit choice on the menu bar.
2. Press the Enter key to display the Edit pull-down menu, and select the Create option.
3. Press the Enter key to display the Create action window.
4. Type the name of the channel in the field provided.

Figure 52. The Message Channel List panel pull-down menus
5. Select the channel type for this end of the link.
6. Press the Enter key.

Notes:
1. If you are using distributed queuing without CICS as well, don’t use any of the same channel names.
2. You are recommended to name all the channels in your network uniquely. As shown in Table 1 on page 30, including the source and target queue manager names in the channel name is a good way to do this.

You are presented with the appropriate Settings panel for the type of channel you have chosen. Fill in the fields with the information you have gathered previously, and select the Save option from the Channel pull-down menu.

You are provided with help in deciding on the content of the various fields in the descriptions of the channel definition panels in the following sections of this chapter.

**Altering a channel**

To alter an existing channel, begin at the Message Channel List panel:

1. Select a channel.
2. Press function key F10 and move the cursor to the Edit choice on the menu bar.
3. Press the Enter key to display the Edit pull-down menu, and select the Alter option.

You are presented with the appropriate Settings panel for the channel you have chosen. Alter the fields with the information you have gathered previously, and select the Save option from the Channel pull-down menu.

You are provided with help in deciding on the content of the various fields in the descriptions of the channel definition panels in the following sections of this chapter, and in the contextual help panels.

**Browsing a channel**

To browse the settings of a channel, begin at the Message Channel List panel:

1. Select a channel.
2. Press the Enter key.

If you try to select more than one channel, only the first one you select is valid.

This results in the respective Settings panel being displayed with details of the current settings for the channel, but with the fields protected against user input.

If the Channel pull-down menu is selected from the menu bar, the Save option is unavailable and this is indicated by an asterisk (*) in place of the first letter, as shown in Figure 53 on page 389.
Renaming a channel

To rename a message channel, begin at the Message Channel List panel:
1. Ensure that the channel is inactive.
2. Select the channel.
3. Use **Copy** to create a duplicate with the new name.
4. Use **Delete** to delete the original channel.

If you decide to rename a message channel, ensure that both ends of the channel are renamed at the same time.

Selected menu-bar choice

The options available in the Selected pull-down menu are:

<table>
<thead>
<tr>
<th>Menu option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Start</td>
<td>Starts the selected channel.</td>
</tr>
<tr>
<td>Stop</td>
<td>Requests the channel to close down, immediately, or controlled.</td>
</tr>
<tr>
<td>Resync</td>
<td>Requests the channel to re-synchronize with the remote end, and then close. No messages are sent.</td>
</tr>
<tr>
<td>Reset</td>
<td>Requests the channel to reset the sequence numbers on this end of the link. The numbers must be equal at both ends for the channel to start.</td>
</tr>
<tr>
<td>Resolve</td>
<td>Requests the channel to resolve in doubt messages without establishing connection to the other end.</td>
</tr>
<tr>
<td>Display Status</td>
<td>Displays the current status of the channel.</td>
</tr>
<tr>
<td>Display Settings</td>
<td>Displays the current settings for the channel.</td>
</tr>
<tr>
<td>Ping</td>
<td>Exchanges a data message with the remote end.</td>
</tr>
<tr>
<td>Exit</td>
<td>Exits from the program.</td>
</tr>
</tbody>
</table>

Start

The **Start** option is available for sender and requester channels, and moreover should not be necessary where a sender channel has been set up with queue manager triggering. For the method of setting up triggering, see "How to trigger channels" on page 390.

When a server channel has been fully defined as a sender, then the same applies as for sender channels.
When you choose the Start option, an EXEC CICS START call is issued to the MCA, which reads the channel definition file and opens the transmission queue. A channel startup sequence is executed which remotely starts the corresponding MCA of the receiver or server channel. When they are running, the sender and server processes await messages arriving on the transmission queue and transmit them as they arrive.

Using the Start option always causes re-synchronization where necessary.

For the start to succeed:
• Channel definitions, local and remote must exist.
• The associated transmission queue must exist and it must be enabled for GETs. If sequential numbering is required, then no other process can have the transmission queue open for input.
• CICS transactions, local (and remote if it is z/OS using CICS) must exist.
• CICS communication must be running.
• The queue managers must be running, local and remote.
• Channel must be inactive.
• Sequence number queue must exist on the receiving system (if it is z/OS using CICS).

It is not necessary that:
• Messages be available
• Remote queue definitions be used
• Remote destination queues be available

A message is returned to the panel confirming that the request to start a channel has been accepted. For confirmation that the start command has succeeded, check the system console for the CICS system hosting the MCA, or the transient data queue.

The sender, server, and requester channel transactions can be started automatically by CICS, if necessary. This is achieved by arranging for the MCA CICS transaction to be started by the CICS system in the required way. This is similar to the triggering startup in that the MCA is passed the required information in a trigger message. For example, it can be customized to start at a certain time every day, or at regular intervals. When started, it retrieves its channel definition and responds accordingly.

How to trigger channels: If triggering is to be used to start a channel when messages arrive on the associated transmission queue, use WebSphere MQ for z/OS operations and control panels or MQSC commands to set it up in accordance with the details on triggering in the WebSphere MQ Application Programming Guide after having collected all the planning data.

Trigger control is exercised by means of the trigger control parameter in the transmission queue definition. You need to set up the transmission queue for the channel, specifying TRIGGER, define an initiation queue, and define a process. For example:

```
DEFINE QLOCAL(MYXMITQ) USAGE(XMITQ) TRIGGER INITQ(MYINITQ) + TRIGTYPE(FIRST) PROCESS(MYPROCESS)
```
DEFINE QLOCAL(MYINITQ)

DEFINE PROCESS(MYPROCESS) APPLTYPE(CICS) APPLICID(CKSG) +
USERDATA(MYCHANNEL)

On the process definition:

**APPLICID**

Names the application that is to be triggered. If you have a fully defined server channel (see “Message channels” on page 8), this ID should be CKSG rather than CKSV. CKSV should be used only for requester-server channels that are to be initiated only by the requester.

**APPLTYPE**

Specifies that this is a CICS application.

**USERDATA**

Specifies the name of the sender channel to be started.

Following the definitions, the long-running trigger process, CKTI, must be started to monitor the initiation queue:

```
CKQC STARTCKTI MYINITQ
```

CKTI waits for trigger messages from the initiation queue, and starts an instance of CKSG for the sender channel in response to the trigger messages. If the channel experiences problems, the trigger control parameter on the transmission queue definition is set to NOTRIGGER by the MCA, and the transmission queue is set to GET(DISABLED). After diagnosis and correction and before you can restart triggering, you must reset the TRIGGER parameter, for example with the WebSphere MQ for z/OS operations and control panels, and must reset the transmission queue to GET(ENABLED).

**Stop**

Use the **Stop** option to request the channel to stop activity.

The **Stop** option presents an action window to allow you to confirm your intention to stop the channel, for all four types of channel. For sender and server channels only, you can select the type of stop you require: IMMEDIATE, or QUIESCE. See Figure 54 on page 392 and Figure 55 on page 393.
Stop immediate: This choice forces the channel to close down immediately, if necessary, without completing the current batch of messages, but an attempt is made to syncpoint with the other end of the channel.

Stop immediate is implemented by setting the channel’s transmission queue to GET DISABLED. This means that if multiple channels are active against a transmission queue, issuing a stop immediate against one of the channels causes all channels to be stopped. You need to reset this queue to GET ENABLED using the WebSphere MQ for z/OS operations and control panels or MQSC commands before you attempt to restart the channels.

For more information, see the “Stopping and quiescing channels” on page 67.
Stop quiesce: This choice requests the channel to close down in an orderly way; the current batch of messages is completed, and the syncpoint procedure is carried out with the other end of the channel.

For more information, see “Stopping and quiescing channels” on page 67. For information about restarting stopped channels, see “Restarting stopped channels” on page 69.

Resync
A message channel is synchronized when there are no in-doubt messages. That is, the sending channel and the receiving channel are agreed on the current unit of work number. The Resync option is valid for sender and server channels, but server channels must be fully defined. The option allows the operator to request the channel to re-synchronize with the remote end by resolving any in-doubt messages.

There is no panel associated with this option.

It is to be used only where the channel is currently inactive and in-doubt messages exist. The channel starts up, resolves the in-doubt messages, and then terminates. It is not intended that the channel should send messages after the resolution has been completed.

If the re-synchronization of a channel is not successful, you may need to examine the content of the system sequence number queue, using the Display Status option from the Selected pull-down menu on the Message Channel List panel. Compare the sequence numbers, or LUWIDs, at the sending and receiving ends of the channel in order to ascertain what needs to be done to restore synchronization.

It may be necessary to reset sequence numbers, or resolve in-doubt message status, if a channel remains out of synchronization.
If a channel terminates abnormally, the sender may be left in doubt as to whether the receiver has received and committed one message, or a batch of messages. When the channel is restarted, the channel program automatically re-synchronizes before sending any new messages.

However, there are times when you may want to re-synchronize the in-doubt messages, but not send any new ones. For example:

- You may want to reset sequence numbers before sending the next batch of messages.
- You may want to close out a batch, but hold the remaining messages for later transmission.

The channel program started by this option establishes a session with a partner. It then exchanges the re-synchronization flows. Then, instead of starting new message traffic, it sends a disconnect flow. The result is that the channel terminates normally, without any in-doubt messages. It is ready to be restarted or reset, as required.

For the re-synchronization to succeed:
- Channel definitions, local and remote must exist
- Transmission queue is available and usable
- CICS transactions, local (and remote if using z/OS with CICS) must exist
- CICS communication must be running
- Queue managers must be running, local and remote
- Sequence number queue must exist on the receiving system (if using z/OS with CICS)
- The channel must be inactive

A message is returned to the panel indicating whether the request to re-synchronize a channel has succeeded. If the Resync process was not successful, check the system console, or transient data queue (TDQ), for the CICS system hosting the MCA for error messages.

**Reset**

Use the **Reset** option to request the channel to reset the sequence number. For a view of the Reset Channel Sequence Number action window, see Figure 56 on page 395. The change must be made separately on each end of the link, with care, and can be done only on inactive channels that have no in-doubt units of work outstanding.

The current sequence number is retrieved and changed to the value requested by the user.

For the reset to succeed:
- The channel sequence number record must exist
- The channel must be inactive
- The channel must not be in doubt
- The channel definition, local, must exist
- CICS transactions, local, must exist
- The CICS system hosting the MCA must be connected to the queue manager

**Notes:**

1. To be effective, the sequence number must be reset in both the sender and the receiver channel definitions. The starting sequence number is not negotiated when a channel starts up, nor is there a default provided. Both ends of a channel definition must have the same sequence number value.
2. In WebSphere MQ for z/OS using CICS, DQM saves the last sequence number sent, which means that to start the next message with sequence number 100, for example, you need to reset the sequence number to 99.

3. If you delete the channel definition at the partner end of the channel (by deleting and recreating the partner queue manager), you must reset the channel sequence number to 0 at the z/OS end and to 1 at the partner end.

Resolve

Use the Resolve option to request a channel to commit or back out in-doubt messages. This may be used when the other end of the link has terminated, and there is no prospect of it returning. Any outstanding units of work need to be resolved with either backout or commit. Backout restores messages to the transmission queue, while Commit discards them.

The Resolve option is needed when the Resync option is not available, or not effective, and messages are held in doubt by a sender or server. The option accepts one of two parameters: Backout or Commit. See Figure 57 on page 396.

The channel program does not try to establish a session with a partner. Instead, it determines the logical unit of work identifier (LUWID) which represents the in-doubt messages. It then issues, as requested, either:

- Backout to restore the messages to the transmission queue; or
- Commit to delete the messages from the transmission queue

For the resolution to succeed:
- The channel must be inactive
- The channel must be in doubt
- The channel type must be sender or server
- The channel definition, local, must exist
- CICS transactions, local, must exist
- Queue manager must be running, local
- The CICS system hosting the MCA must be connected to the queue manager

See “In-doubt channels” on page 70 for more information.
Use the Display Status option to display the current status of the channel. The following information is displayed:

- Whether the channel is active or inactive
- The in-doubt status of sender and server channels
- The sequence number last sent, if sequence numbering is in effect
- The last LUWID number, if available. Available means:
  - Always available for receiver and requester channels
  - Available for sender and server channels when:
    - Sequence numbering is in effect
    - No sequence numbering in effect, but the channel is in doubt

That is, the LUWID number is not available for sender and server channels when sequence numbering is not in effect and the channel is not in doubt.

For an example of sender and server status panels, see Figure 58 on page 397 and for an example of receiver and requester status panels, see Figure 59 on page 397.

‘Not available’ status is acceptable when:

- Showed for a sequence number, if the channel is active
- Showed for an LUWID when the channel is not in doubt

Otherwise, if a ‘Not available’ status is shown in any of the fields, this indicates that an error has occurred, and you should refer to the console log to find the error messages associated with this problem.
Figure 58. An example of a sender channel Display Channel Status window. The server channel Display Channel Status panel looks the same, except that the Channel type field is changed to SERVER.

Figure 59. An example of a receiver channel Display Channel Status window. The requester channel Display Channel Status window looks the same, except that the Channel type field is changed to REQUESTER.

**Display settings**

Use the Display Settings option to display the current definitions for the channel. This choice displays the appropriate panel for the type of channel with the fields displaying the current values of the parameters, and protected against user input:

- Sender: see Figure 70 on page 408
Protected input is shown with colon characters (:) at the end of field descriptions, and the Save option is not available on the Channel pull-down menu.

You can select this choice from the Message Channel List panel by choosing a channel and pressing Enter, without using the menu bar, ensuring that the cursor is not on the menu bar.

### Ping

Use the Ping option to exchange a data message with the remote end. This gives you some confidence that the link is available and functioning. It can be issued from sender and server channels only, but server channels must be fully defined.

Ping does not involve the use of transmission queues and target queues. It uses channel definitions, the related CICS communication link, the network setup, and the queue managers at both ends.

The corresponding channel is started at the far side of the link, and performs the startup parameter negotiation.

If an error occurs, an error message is displayed on the panel, and additional messages may be written to the console, or the CICS transient data queue.

The Ping panel offers you the opportunity to enter a message of up to 20 characters to be exchanged across the link. If you do not make use of this, a default message is used.

The result of the message exchange is presented in the Ping panel for you, and this is the returned message text, together with the time the message was sent, and the time the reply was received.

Installations may supply their own applications to exchange particular information, such as system identifiers. Figure 60 shows a view of the Ping action window.

![Figure 60. The Ping action window](image-url)
Exit
Use the Exit option to exit the current function: channel settings, help, or message channel list.

A secondary window appears when you try to exit a channel settings panel without first saving any changed definitions. This is a safe exit to prevent inadvertent loss of data. The secondary window is shown in Figure 61.

<table>
<thead>
<tr>
<th>Channel</th>
<th>Help</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Save</td>
<td>13.2._VC14.SENDER - Settings</td>
</tr>
<tr>
<td>2. Exit</td>
<td>VICY14</td>
</tr>
<tr>
<td></td>
<td>+----------------------------------</td>
</tr>
<tr>
<td></td>
<td>+----------------------------------</td>
</tr>
<tr>
<td></td>
<td>VCI3.2. VC14.SENDER - Exit</td>
</tr>
<tr>
<td></td>
<td>More: +</td>
</tr>
<tr>
<td></td>
<td>--------------------------------</td>
</tr>
<tr>
<td></td>
<td>Channel type . . . : SENDER</td>
</tr>
<tr>
<td></td>
<td>Target syst</td>
</tr>
<tr>
<td></td>
<td>Transmiss</td>
</tr>
<tr>
<td></td>
<td>Batch size</td>
</tr>
<tr>
<td></td>
<td>Sequence nu</td>
</tr>
<tr>
<td></td>
<td>Max message</td>
</tr>
<tr>
<td></td>
<td>Max transmi</td>
</tr>
<tr>
<td></td>
<td>Disconnect</td>
</tr>
<tr>
<td></td>
<td>Transaction</td>
</tr>
<tr>
<td></td>
<td>Connection</td>
</tr>
<tr>
<td></td>
<td>CICS profile name . . . . . . .</td>
</tr>
</tbody>
</table>

Figure 61. The Exit confirmation secondary window

Edit menu-bar choice
The options available in the Edit pull-down menu are:
- Copy
- Create
- Alter
- Delete
- Find

In any of the action windows and settings panels associated with Edit, you can type the channel name in uppercase or lowercase, but it may be converted to uppercase when you press the Enter key, depending upon your Typeterm definition.

Copy
Use the Copy option to copy an existing channel. The Copy action window (see Figure 62 on page 400) enables you to define the new channel name. You can use the characters shown in “Create” on page 400 in the name.

Press the Enter key on the Copy action window to display the channel settings panel with details of current system values. You can change any of the new channel settings. You save the new channel definition by selecting Channel from the menu bar, and selecting the Save option from the pull-down menu.
Create

Use the Create option to create a new channel definition from a screen of fields filled with default values supplied by WebSphere MQ for z/OS. Figure 63 on page 401 shows you where to type the name of the channel, and how to select the type of channel you are creating.

When you press the Enter key, the appropriate channel settings panel is displayed. Type information in all the necessary fields in this panel and then save the definition by selecting Channel from the menu bar, and selecting the Save option from the pull-down menu.

The channel name must be the same at both ends of the channel, and unique within the network. You can use the following characters in the name:

- Uppercase A-Z
- Lowercase a-z
- Numerics 0-9
- Period '.
- Forward slash '/'
- Underscore '_'
- Percentage sign '%'

Figure 62. The Copy action window
All panels have default values supplied for some fields. You can change the values when you are creating or copying channels. For examples of the channel definition panels showing the default values, see Figure 64.

Press the Enter key on the Create action window to display the channel settings panel with details of default values.

You can create your own set of channel default values by setting up dummy channels with the required defaults for each channel type, and copying them each time you want to create new channel definitions.

Figure 63. The Create action window

Figure 64. Example of default values during Create for a channel. The values supplied cannot be customized.
**Message Channel List panel**

**Alter**
Use the **Alter** option to change an existing channel definition, except for the channel name. Simply type over the fields to be changed in the channel definition panel, and then save the updated definition by selecting **Channel** from the menu bar, and selecting the **Save** option from the pull-down menu.

**Delete**
Use the **Delete** option to delete the selected channel. For the secondary window requesting confirmation of your intention, see **Figure 65**.

![Figure 65. The Delete action window](image)

**Find**
Use the **Find** option to locate a particular channel name from the list of available channels. If the name of the channel you want is found, it is placed at the top of the list on the Message Channel List panel. The Find a Channel action window is shown in **Figure 66**.

![Figure 66. The Find a Channel action window](image)

You can partially define the channel name using a terminating asterisk, for example, `channel.lon*`. This results in the first channel name to be found with these initial letters being placed at the top of the list.
View menu-bar choice

The options available in the View pull-down menu change the current view of the list shown on the Message Channel List panel; see Figure 67.

Menu option

Include all

All channels are included in the list.

Include...

Select the channels to be included in the list, by means of an action window.

You can partially define the channel name using a terminating asterisk, for example, channel.lon*. This results in channel names found with these initial letters being included in the list.

Also in the action window is a field to allow you to specify a channel type, or all types of channel.

Refresh now F5

Updates the panel with fresh data from the system.

Figure 67. The Include search criteria action window
Help menu-bar choice

The Help pull-down menu is shown in Figure 68.

Table 37. Menu-bar choices on channel panels

<table>
<thead>
<tr>
<th>Channel name</th>
<th>Sysid</th>
<th>Type</th>
<th>Help</th>
</tr>
</thead>
<tbody>
<tr>
<td>VC13.2.VC14.JAC3</td>
<td>VR14</td>
<td>REQUESTER</td>
<td>2. General help</td>
</tr>
</tbody>
</table>

Figure 68. The Help pull-down menu

The channel definition panels

The four channel Settings panels for defining channels (one for each of sender, receiver, server, and requester) have a menu bar with choices you can pull down to reveal various options you can select for these choices. See Table 37.

The menu-bar choices are:

Table 37. Menu-bar choices on channel panels

<table>
<thead>
<tr>
<th>Channel name</th>
<th>Sysid</th>
<th>Type</th>
<th>Help</th>
</tr>
</thead>
<tbody>
<tr>
<td>VC13.2.VC14.JAC3</td>
<td>VR14</td>
<td>REQUESTER</td>
<td>2. General help</td>
</tr>
</tbody>
</table>

The work area of the panels is used to present the fields of attributes or settings for the channel.

The function keys control the use of the panels to:

- Call help panels
- Move the cursor to the menu bar
- Refresh the panel
- Cancel a pull-down menu or a secondary window
- Exit from the panel
- Scroll forward and backward through settings

The method of using the panels is:

- For new channels, fill in the data fields, then select Channel from the menu bar, and select the Save option from the pull-down menu.

Note: Default values supplied by WebSphere MQ for z/OS are presented in some fields. The defaults cannot be changed, but the values presented can be changed.

- For existing channels, type over the data presented in the fields with new data. Then select Channel from the menu bar, and select the Save option from the pull-down menu.
Channel menu-bar choice

The Channel menu-bar choice enables you to save any changes you have made to channel definitions, and to return to the Message Channel List panel.

Saving changes

If there are no errors, selecting the Save option from the Channel pull-down menu saves any changes you have made to channel definitions. You are returned to the Message Channel List panel.

If there are errors, you are returned to the Settings panel with an error message, and all fields containing errors are highlighted. The cursor is positioned on the first field in error. The changes are not saved.

Exit from the panel

Selecting the Save option from the Channel pull-down menu saves the changes you have made and returns you to the Message Channel List panel.

Selecting the Exit option from the Channel pull-down menu, or pressing F3 or F12, returns you to the Message Channel List panel.

However, if you have not saved the changes you made, a secondary window requesting confirmation of your intention to exit without saving the data is presented; see Figure 61 on page 399. If you want to save the changes you have made, select Save and exit. If you have had second thoughts about the changes you have made, select Exit without saving.

Help menu-bar choice

The Help pull-down menu is shown in Figure 69.

```
<table>
<thead>
<tr>
<th>Channel</th>
<th>Help</th>
<th>Settings</th>
<th>CICS01</th>
</tr>
</thead>
<tbody>
<tr>
<td>MCATTB1</td>
<td>1. Using help</td>
<td>- Settings</td>
<td>CICS01</td>
</tr>
<tr>
<td></td>
<td>2. General help</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>3. Keys help</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>4. Tutorial</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>5. Product Info</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

Figure 69. The Help choice pull-down menu
Channel settings panel fields

The fields in these panels define the attributes of the channels. The channel settings panel fields that you can change are shown in Table 38. You can find details for each field in Chapter 6, “Channel attributes” on page 77.

A “✓” signifies that the field is available for use with the indicated type of channel, while an “O” means that these fields are only needed for server channels when they are to be used as sender channels.

**Table 38. Channel attribute fields per channel type**

<table>
<thead>
<tr>
<th>Attribute field</th>
<th>Sender</th>
<th>Server</th>
<th>Receiver</th>
<th>Requester</th>
</tr>
</thead>
<tbody>
<tr>
<td>Batch size</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>CICS profile name</td>
<td>✓</td>
<td>O</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Connection name</td>
<td>✓</td>
<td>O</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Disconnect interval</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>LU62 TP name (see Note)</td>
<td>✓</td>
<td>O</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Maximum message size</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Maximum transmission size</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Message exit</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>PUT authority</td>
<td></td>
<td></td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Retry count</td>
<td>✓</td>
<td>O</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Retry fast interval</td>
<td>✓</td>
<td>O</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Retry slow interval</td>
<td>✓</td>
<td>O</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Receive exit</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Sequence number wrap</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Sequential delivery</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Security exit</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Send exit</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Target system identifier</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Transmission queue name</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Transaction identifier</td>
<td>✓</td>
<td>✓</td>
<td>O</td>
<td></td>
</tr>
</tbody>
</table>

**Note:** See also the *Multiplatform APPC Configuration Guide* (“Red Book”) and Table 39 for information.

**Table 39. Settings for LU 6.2 TP name on the local z/OS system for a remote queue manager platform**

<table>
<thead>
<tr>
<th>Remote platform</th>
<th>Sender/server</th>
<th>Requester</th>
</tr>
</thead>
<tbody>
<tr>
<td>z/OS using CICS</td>
<td>CKRC</td>
<td>CKSV¹</td>
</tr>
<tr>
<td>z/OS without CICS and UNIX systems</td>
<td>As specified in the side information on remote queue manager system</td>
<td>As specified in the side information on remote queue manager system</td>
</tr>
<tr>
<td>OS/2</td>
<td>As specified in the OS/2 Run Listener command, or defaulted from the OS/2 queue manager configuration file</td>
<td>As specified in the OS/2 Run Listener command, or defaulted from the OS/2 queue manager configuration file</td>
</tr>
</tbody>
</table>
Table 39. Settings for LU 6.2 TP name on the local z/OS system for a remote queue manager platform (continued)

<table>
<thead>
<tr>
<th>Remote platform</th>
<th>Sender/server</th>
<th>Requester</th>
</tr>
</thead>
<tbody>
<tr>
<td>OS/400</td>
<td>The same as the compare value in the routing entry on the OS/400 system</td>
<td>The same as the compare value in the routing entry on the OS/400 system</td>
</tr>
<tr>
<td>Digital OVMS</td>
<td>As specified in the Digital OVMS Run Listener command</td>
<td>As specified in the Digital OVMS Run Listener command</td>
</tr>
<tr>
<td>Compaq NonStop Kernel</td>
<td>The same as the TPNAME specified in the receiver-channel definition</td>
<td>The same as the TPNAME specified in the receiver-channel definition</td>
</tr>
<tr>
<td>Windows NT</td>
<td>As specified in the Windows NT Run Listener command, or the invokable Transaction Program that was defined using TpSetup on Windows NT</td>
<td>As specified in the Windows NT Run Listener command, or the invokable Transaction Program that was defined using TpSetup on Windows NT</td>
</tr>
</tbody>
</table>

Note: 1 If you have a fully defined server channel, (see "Message channels" on page 8), its definition should specify a transaction ID of CKSG.

If you have more than one queue manager on the same machine, ensure that the TPnames in the channel definitions are unique. To modify a TPname, use CSQ4SIDE or CKMC.
Details of sender channel settings panel

This section provides details of the sender channel settings panel, as shown in Figures 70 and 71.

Channel settings panel fields

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Channel type</td>
<td>SENDER</td>
</tr>
<tr>
<td>Target system id</td>
<td></td>
</tr>
<tr>
<td>Transmission queue name</td>
<td>TX1</td>
</tr>
<tr>
<td>Batch size</td>
<td>0001</td>
</tr>
<tr>
<td>Sequence number wrap</td>
<td>0999999</td>
</tr>
<tr>
<td>Max message size</td>
<td>0032000</td>
</tr>
<tr>
<td>Max transmission</td>
<td>32000</td>
</tr>
<tr>
<td>Disconnect interval</td>
<td>0001</td>
</tr>
<tr>
<td>Transaction id</td>
<td>CKSG</td>
</tr>
<tr>
<td>Connection name</td>
<td>HtoH</td>
</tr>
<tr>
<td>CICS profile name</td>
<td></td>
</tr>
<tr>
<td>LU 6.2 TP name</td>
<td>CKRC</td>
</tr>
</tbody>
</table>

Figure 70. The sender channel settings panel

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Channel type</td>
<td>SENDER</td>
</tr>
<tr>
<td>Sequential delivery</td>
<td>0 (0=No or 1=Yes)</td>
</tr>
<tr>
<td>Retry</td>
<td></td>
</tr>
<tr>
<td>Count</td>
<td>005</td>
</tr>
<tr>
<td>Fast interval</td>
<td>005</td>
</tr>
<tr>
<td>Slow interval</td>
<td>030</td>
</tr>
<tr>
<td>Exit routines</td>
<td></td>
</tr>
<tr>
<td>Security</td>
<td></td>
</tr>
<tr>
<td>Message</td>
<td></td>
</tr>
<tr>
<td>Send</td>
<td></td>
</tr>
<tr>
<td>Receive</td>
<td></td>
</tr>
</tbody>
</table>

Figure 71. The sender channel settings panel - screen 2
Details of receiver channel settings panel

This section provides details of the receiver channel settings panels, as shown in Figures 72 and 73.

```
Channel  Help
----------------------------------------------
MCATB3   VICY13.TO.VICY14 - Settings          VICY14

Channel type ........: RECEIVER
Target system id .....:
Batch size ..........: 0100
Sequence number wrap : 0099920
Max message size ... : 0032000
Max transmission ... : 32000

F1=Help  F3=Exit  F5=Refresh now  F7=Bkwd  F8=Fwd  F10=Menu Bar
F12=Cancel
```

Figure 72. The receiver channel settings panel

```
Channel  Help
----------------------------------------------
MCATC3   VICY13.TO.VICY14 - Settings          VICY14

Type information. Then select an action.                          More: -
Channel type ........: RECEIVER
Sequential delivery ...: 1  (0=No or 1=Yes)
Put authority .......: 1  (1=Process or 2=Context)

Exit routines
   Security ...........
   Message ...........
   Send .............
   Receive ..........

F1=Help  F3=Exit  F5=Refresh now  F7=Bkwd  F8=Fwd  F10=Menu Bar
F12=Cancel
```

Figure 73. The receiver channel settings panel - screen 2
### Channel settings panel fields

**Details of server channel settings panel**

This section provides details of the server channel settings panels, as shown in Figures 74 and 75.

<table>
<thead>
<tr>
<th>Channel</th>
<th>Help</th>
</tr>
</thead>
<tbody>
<tr>
<td>MCATTB1</td>
<td>HURSLEY.TO.SYDNEY - Settings</td>
</tr>
</tbody>
</table>

---

**Channel type**: SERVER

<table>
<thead>
<tr>
<th>Target system id</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Transmission queue name</th>
</tr>
</thead>
<tbody>
<tr>
<td>TX1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Batch size</th>
</tr>
</thead>
<tbody>
<tr>
<td>0001</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Sequence number wrap</th>
</tr>
</thead>
<tbody>
<tr>
<td>0999999</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Max message size</th>
</tr>
</thead>
<tbody>
<tr>
<td>0032000</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Max transmission</th>
</tr>
</thead>
<tbody>
<tr>
<td>32000</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Disconnect interval</th>
</tr>
</thead>
<tbody>
<tr>
<td>0001</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Transaction id</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Connection name</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CICS profile name</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>LU 6.2 TP name</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
</tr>
</tbody>
</table>

---

**Figure 74. The server channel settings panel**

<table>
<thead>
<tr>
<th>Channel</th>
<th>Help</th>
</tr>
</thead>
<tbody>
<tr>
<td>MCATTC1</td>
<td>HURSLEY.TO.SYDNEY - Settings</td>
</tr>
</tbody>
</table>

---

**Channel type**: SERVER

<table>
<thead>
<tr>
<th>Sequential delivery</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 (0=No or 1=Yes)</td>
</tr>
</tbody>
</table>

---

**Figure 75. The server channel settings panel - screen 2**
Details of requester channel settings panel

This section provides details of each field in the requester channel settings panels, as shown in Figures 76 and 77.

```
Channel Help
--------------------------------------------
MCATB4 VICY13.TO.VICY14.CB - Settings VICY14

More: +

Channel type .......: REQUESTER
Target system id ...:
Batch size .........: 0001
Sequence number wrap.: 0999999
Max message size .....: 0032000
Max transmission ....: 32000
Transaction id .....: CKRQ
Connection name.....: VC13
CICS profile name...: LU6PROF
LU 6.2 TP name .....: CKSV

F1=Help F3=Exit F5=Refresh now F7=Bkwd F8=Fwd F10=Menu Bar
F12=Cancel
```

Figure 76. The requester channel settings panel

```
Channel Help
--------------------------------------------
MCATTC4 VICY13.TO.VICY14.CB - Settings VICY14

More: -

Channel type .......: REQUESTER
Sequential delivery ...: 0 (0=No or 1=Yes)
Put authority .......: 1 (1=Process or 2=Context)
Retry
   Count ..........: 005
   Fast interval ..: 005
   Slow interval ..: 030
Exit routines
Security ..........:
Message ..........:
Send .............:
Receive ..........:

F1=Help F3=Exit F5=Refresh now F7=Bkwd F8=Fwd F10=Menu Bar
F12=Cancel
```

Figure 77. The requester channel settings panel - screen 2
Channel settings panel fields
Chapter 29. Preparing WebSphere MQ for z/OS when using CICS

This chapter describes the WebSphere MQ for z/OS and CICS preparations you need to make before you can start to use CICS for distributed queuing.

To enable distributed queuing, you must perform the following three tasks:

- Customize the distributed queuing facility and define the WebSphere MQ objects required; this is described in WebSphere MQ for z/OS Concepts and Planning Guide and WebSphere MQ for z/OS System Setup Guide.
- Define access security; this is described in WebSphere MQ for z/OS Concepts and Planning Guide and WebSphere MQ for z/OS System Setup Guide.
- Set up your communications; this is described in Chapter 30, “Setting up CICS communication for WebSphere MQ for z/OS” on page 417.

Defining DQM requirements to WebSphere MQ

In order to define your distributed-queuing requirements, you need to:

- Define WebSphere MQ programs and data sets as CICS resources
- Define the channel definitions
- Define the CKMQ transient data queue
- Define WebSphere MQ queues triggers and processes
- Define CICS resources used by distributed queuing
- Define access security

See the WebSphere MQ for z/OS System Setup Guide for information about these tasks.

Defining WebSphere MQ objects

Use the WebSphere MQ for z/OS operations and control panels, or one of the other WebSphere MQ for z/OS command input methods, to define WebSphere MQ for z/OS objects. Refer to the WebSphere MQ Script (MQSC) Command Reference book for details of defining objects.

You define:

- A local queue with the usage of (XMITQ) for each sending message channel.
- Remote queue definitions.

A remote queue object has three distinct uses, depending upon the way the name and content are specified:

- Remote queue definition
- Queue manager alias definition
- Reply-to queue alias definition

This is shown in Table 2 on page 37.

- A process naming the MCA sender transaction, CKSG, as the application to be triggered by messages appearing on the transmission queue. The process definition parameter, USERDATA, must contain the name of the channel to be started by this process. See “How to trigger channels” on page 390.

The supplied sample CSQ4DISQ gives examples of the necessary definitions.
Defining WebSphere MQ objects

Multiple message channels per transmission queue

It is possible to define more than one channel per transmission queue, but only one of these channels needs to be active at any one time. The provision of multiple channels is recommended to provide alternative routes between queue managers for traffic balancing and link failure recovery.

You may start more than one channel to serve a transmission queue to increase message throughput, but when doing so, ensure that the queue has a SHARE attribute, and that there is not a need for sequential delivery of messages.
Channel operation considerations

Channels are designed to be active only when there is work for them to process. This mechanism allows for conservation of limited system resources such as active transactions and LU 6.2 sessions while at the same time delivering messages in a timely fashion determined by the application. The mechanisms which are used to determine when a channel is started and stopped are triggering and the disconnect interval respectively.

This mechanism works well unless the operator wishes to terminate a channel before the disconnect time interval expires. This can occur in the following situations:

- System quiesce
- Resource conservation
- Unilateral action at one end of a channel

In these cases it is necessary to stop the channel using the STOP option from the Message Channel List panel of the CKMC transaction. For information about what happens when a channel is stopped in this way, and how to restart the channel, see “Stopping and quiescing channels” on page 67.
Chapter 30. Setting up CICS communication for WebSphere MQ for z/OS

Distributed queue management (DQM) provides channel control programs which form the interface to CICS communication links, controllable by the system operator. The channel definitions held by DQM use these CICS connections.

When a channel is started, it tries to use the CICS connection specified in the channel definition. For this to succeed, it is necessary for the CICS connection to be defined and available. This section explains how to do this.

If more than one CICS system is associated with any one WebSphere MQ for z/OS, and each CICS system is running some DQM functions, you need to define connections between the CICS systems. This chapter also explains how to do this.

Connecting CICS systems

Part of the installation of DQM requires the definition and installation of CICS logical unit type 6.2 (LU 6.2) connections that provide the physical link between the CICS systems serving the local queue manager, and the systems serving the remote queue managers. To set up these connections, use the CICS Intercommunication Guide.

One z/OS system can be host to a number of CICS systems at the same time, and each CICS system is able to connect to one queue manager at any one time.

You provide communication links so that queue managers may use these links, through CICS intersystem communication (ISC) to reach other queue managers on z/OS systems (using CICS or not), and on other non-z/OS systems, provided they are using the standard queue manager intercommunication protocol, WebSphere MQ Message Channel Protocol.

Communication between queue managers

There are two forms of communication between CICS systems:

- Intersystem communication (ISC): communication between a CICS system and other systems in a data communication network that support the logical unit type 6.1 or logical unit type 6.2 protocols of IBM Systems Network Architecture (SNA).
- Multiregion operation (MRO): communication between CICS systems running in different address spaces of the same z/OS system.

Only ISC LU 6.2 protocols are used for connecting two queue managers over a DQM channel, even where they both reside in the same z/OS system.

Note: CICS for MVS/ESA Version 4 Release 1.0 or higher is required for WebSphere MQ distributed queue management.

Intersystem communication

The connection type must be ISC LU 6.2, but can be defined as one of the following:

- LU 6.2 single-session terminal
Preparation on z/OS using CICS

- LU 6.2 single-session connection
- LU 6.2 parallel-session connection

Before deciding the type of connection to be defined, you should consider the following points:

- The number of channels to be defined between the two systems
- The maximum number of channels that are to be active at any one time
- How often the connection is used
- The number of channels per transmission queue
- The number of channels that can be active per connection

**Note:** Multiple channels can be active on the same connection.

To define an LU 6.2 link between the two CICS systems, you should refer to the following books:


paying particular attention to the sections discussing communication resources.

### Defining an LU 6.2 connection

When you decide which type of LU 6.2 connection is to be established between the local and remote CICS systems, the process of definition can take place.

Only one ISC connection can be active between any two CICS systems at the same time. However, a single CICS system can have connections to multiple remote CICS systems at the same time.

The sender and requester channel definitions require the provision of the LU 6.2 connection name and, optionally, the CICS profile name to be used.

The relationship between CICS profiles and connections is shown in Figure 78. The uppercase fields are the names of the CEDA transaction entry, and the lowercase values are fields within those definitions that are relevant to the example.

![Figure 78. CICS LU 6.2 connection definition](image)

If a sender channel is defined with the following characteristics, it causes a session to be allocated using a SES1 session on connection CON1:

- `CHANNEL=MY.CHANNEL`
- `CONNECTION NAME=CON1`
- `CICS PROFILE NAME=MYPROF`

If no CICS profile name is specified in the channel definition, DQM does not specify a profile when allocating a session.
Installing the connection

When you have defined the connection definitions on your CICS system definitions (CSDs), these can be installed using the CICS CEDA INSTALL command.

If you want to install these connections as part of the CICS initialization process, you can add the group that contains the connection definitions to the CICS startup list that is specified in the GRPLIST= parameter. You then need to cold start your CICS system for the entries to become effective.

Communications between CICS systems attached to one queue manager

DQM functions may be shared between more than one CICS system. When these CICS systems are connected to, or associated with, the same queue manager, then these CICS systems need to be set up correctly so that function shipping of EXEC CICS commands and program invocation occur correctly.

Connection names for function shipping

Although CICS does not require that a connection name is the same as the DFHSIT SYSIDNT name of the target CICS system, DQM requires that they are the same.

The type of connection can be either MRO or ISC.
Preparation on z/OS using CICS
Chapter 31. Example configuration - IBM WebSphere MQ for z/OS using CICS

This chapter gives an example of how to set up communication links using CICS from WebSphere MQ for z/OS or MQSeries for OS/390 or MVS/ESA to WebSphere MQ products on the following platforms:

- OS/2
- Windows
- AIX
- HP-UX
- AT&T GIS UNIX®
- iSeries

(You can also connect any of the following:
  - z/OS to z/OS
  - z/OS to MVS/ESA
  - MVS/ESA to MVS/ESA
with CICS.)

First it describes the parameters needed for an LU 6.2 connection; then it describes:

- "Establishing an LU 6.2 connection using CICS" on page 424

Once the connection is established, you need to define some channels to complete the configuration. This is described in "WebSphere MQ for z/OS using CICS configuration" on page 426.

See Chapter 7, "Example configuration chapters in this book" on page 101 for background information about this chapter and how to use it.

Configuration parameters for an LU 6.2 connection

Table 40 on page 422 presents a worksheet listing all the parameters needed to set up communication from z/OS to one of the other WebSphere MQ platforms. The worksheet shows examples of the parameters, which have been tested in a working environment, and leaves space for you to fill in your own values. An explanation of the parameter names follows the worksheet. Use the worksheet in this chapter in conjunction with the worksheet in the chapter for the platform to which you are connecting.
The steps required to set up an LU 6.2 connection are described in "Establishing an LU 6.2 connection using CICS" on page 424, with numbered cross references to the parameters on the worksheet.

Configuration worksheet

Use this worksheet to record the values you use for your configuration. Where numbers appear in the Reference column they indicate that the value must match that in the appropriate worksheet elsewhere in this book. The examples that follow in this chapter refer back to the values in the ID column. The entries in the Parameter Name column are explained in "Explanation of terms" on page 423.

Table 40. Configuration worksheet for z/OS using LU 6.2

<table>
<thead>
<tr>
<th>ID</th>
<th>Parameter Name</th>
<th>Reference</th>
<th>Example Used</th>
<th>User Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Command prefix</td>
<td>+cpf</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Network ID</td>
<td>NETID</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Node name</td>
<td>MVSPU</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Local LU name</td>
<td>MVSLU</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Symbolic destination</td>
<td>M1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>Modename</td>
<td>#INTER</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>Local Transaction Program name</td>
<td>MQSERIES</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>LAN destination address</td>
<td>400074511092</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Connection to an OS/2 system using CICS

The values in this section of the table must match those used in Table 15 on page 144, as indicated.

<table>
<thead>
<tr>
<th>ID</th>
<th>Parameter Name</th>
<th>Reference</th>
<th>Example Used</th>
<th>User Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>17</td>
<td>Connection name</td>
<td>OS2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>Group name</td>
<td>EXAMPLE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>Session name</td>
<td>OS2SESS</td>
<td></td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>Netname</td>
<td>OS2LU</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Connection to a Windows system using CICS

The values in this section of the table must match those used in Table 17 on page 168, as indicated.

<table>
<thead>
<tr>
<th>ID</th>
<th>Parameter Name</th>
<th>Reference</th>
<th>Example Used</th>
<th>User Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>17</td>
<td>Connection name</td>
<td>WNT</td>
<td></td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>Group name</td>
<td>EXAMPLE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>Session name</td>
<td>WNTSESS</td>
<td></td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>Netname</td>
<td>WINNTLU</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Connection to an AIX system using CICS

The values in this section of the table must match those used in Table 21 on page 197, as indicated.

<table>
<thead>
<tr>
<th>ID</th>
<th>Parameter Name</th>
<th>Reference</th>
<th>Example Used</th>
<th>User Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>17</td>
<td>Connection name</td>
<td>AIX</td>
<td></td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>Group name</td>
<td>EXAMPLE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>Session name</td>
<td>AIXSESS</td>
<td></td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>Netname</td>
<td>AIXLU</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Connection to an HP-UX system using CICS

The values in this section of the table must match those used in Table 24 on page 219, as indicated.

<table>
<thead>
<tr>
<th>ID</th>
<th>Parameter Name</th>
<th>Reference</th>
<th>Example Used</th>
<th>User Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>17</td>
<td>Connection name</td>
<td>HPUX</td>
<td></td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>Group name</td>
<td>EXAMPLE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>Session name</td>
<td>HPUXSESS</td>
<td></td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>Netname</td>
<td>HPUXLU</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Table 40. Configuration worksheet for z/OS using LU 6.2 (continued)

<table>
<thead>
<tr>
<th>ID</th>
<th>Parameter Name</th>
<th>Reference</th>
<th>Example Used</th>
<th>User Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>17</td>
<td>Connection name</td>
<td>GIS</td>
<td></td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>Group name</td>
<td>EXAMPLE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>Session name</td>
<td>GISSESS</td>
<td></td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>Netname</td>
<td>GISLU</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Connection to an AT&T GIS UNIX system using CICS

The values in this section of the table must match those used in Table 26 on page 243 as indicated.

Connection to an iSeries system using CICS

The values in this section of the table must match those used in Table 48 on page 521 as indicated.

Explanation of terms

1 Command prefix
   This is the unique command prefix of your WebSphere MQ for z/OS queue-manager subsystem. The z/OS systems programmer defines this at installation time, in SYS1.PARMLIB(IEFSSNss), and will be able to tell you the value.

2 Network ID
   The VTAM startup procedure in your installation is partly customized by the ATCSTRxx member of the data set referenced by the DDNAME VTAMLST. The Network ID is the value specified for the NETID parameter in this member. For Network ID you must specify the name of the NETID that owns the WebSphere MQ communications subsystem (WebSphere MQ channel initiator or CICS for z/OS as the case may be). Your network administrator will tell you the value.

3 Node name
   VTAM, being a low-entry network node, does not have a Control Point name for Advanced Peer-to-Peer Networking (APPN) use. It does however have a system services control point name (SSCPNAME). For node name, you must specify the name of the SSCP that owns the WebSphere MQ communications subsystem (WebSphere MQ channel initiator or CICS for z/OS as the case may be). This is defined in the same ATCSTRxx member as the Network ID. Your network administrator will tell you the value.

4 Local LU name
   A logical unit (LU) is software that serves as an interface or translator between a transaction program and the network. It manages the exchange of data between transaction programs. The local LU name is the unique VTAM APPLID of this WebSphere MQ subsystem. Your network administrator will tell you this value.

5 Symbolic destination
   This is the name you give to the CPI-C side information profile. You need a side information entry for each LU 6.2 listener.

6 Modename
   This is the name given to the set of parameters that control the LU 6.2 conversation. An entry with this name and similar attributes must be
defined at each end of the session. In VTAM, this corresponds to a mode table entry. You network administrator will assign this to you.

7 Transaction Program name
WebSphere MQ applications trying to converse with this queue manager will specify a symbolic name for the program to be run at the receiving end. This will have been specified in the TPNAME attribute on the channel definition at the sender. For simplicity, wherever possible use a transaction program name of MQSERIES, or in the case of a connection to VSE/ESA, where the length is limited to 4 bytes, use MQTP.

See [Table 32 on page 363](#) for more information. If the receiving end is z/OS using CICS, special values are required.

8 LAN destination address
This is the LAN destination address that your partner nodes will use to communicate with this host. When you are using a 3745 network controller, it will be the value specified in the LOCADD parameter for the line definition to which your partner is physically connected. If your partner nodes use other devices such as 317X or 6611 devices, the address will have been set during the customization of those devices. Your network administrator will tell you this value.

17 Connection name
(CICS only) This is a 4-character name by which each connection will be individually known in CICS RDO.

18 Group name
(CICS only) You choose your own 8-character name for this value. Your system may already have a group defined for connections to partner nodes. Your CICS administrator will give you a value to use.

19 Session name
(CICS only) This is an 8-character name by which each group of sessions will be individually known. For clarity we use the connection name, concatenated with ‘SESS’.

20 Netname
(CICS only) This is the LU name of the WebSphere MQ queue manager on the system with which you are setting up communication.

### Establishing an LU 6.2 connection using CICS

**Note:** This example is for a connection to an OS/2 system. The steps are the same whatever platform you are using; change the values as appropriate.

**Defining a connection**

1. At a CICS command line type:

   ```
   CEDA DEF CONN(connection name) GROUP(group name) 17
   ```

   For example:
   ```
   CEDA DEF CONN(OS2) GROUP(EXAMPLE)
   ```

2. Press Enter to define the connection to CICS.
   A panel is displayed, as shown below.
3. On this panel, change the **Netname** field in the **CONNECTION IDENTIFIERS** section to be the LU name (20) of the target system. In the **CONNECTION PROPERTIES** section set the **Accessmethod** field to **Vtam** and the **Protocol** to **Appc**.

4. Press Enter to make the change.

### Defining the sessions

1. At a CICS command line type:

   ```
   CEDA DEF SESS(session name) GROUP(group name)
   ```

   For example:

   ```
   CEDA DEF SESS(OS2SESS) GROUP(EXAMPLE)
   ```

2. Press Enter to define the group of sessions for the connection. A panel is displayed, as shown below.
3. On this panel, in the SESSION IDENTIFIERS section, specify the Connection name (SF58000017/SF590000) in the Connection field and set the Modename to #INTER. In the SESSION PROPERTIES section set the Protocol to Appc and the Maximum field to 008, 004.

4. Press Enter to make the change.

**Installing the new group definition**

To install the new group definition, type:

CEDA INS GROUP(group name)

at a CICS command line, and press Enter.

**Note:** If this connection group is already in use, severe errors will be reported. If this occurs you must take the existing connections out of service, retry the group installation, and then set the connections in service again using the following commands:

1. CEMT I CONN
2. CEMT S CONN(*) OUTS
3. CEDA INS GROUP(Group name)
4. CEMT S CONN(*) INS

**What next?**

The connection is now established. You are ready to complete the configuration. Go to **WebSphere MQ for z/OS using CICS configuration**

**WebSphere MQ for z/OS using CICS configuration**

WebSphere MQ channels will not initialize successfully if the channel negotiation detects that the message sequence number is different at each end. You may need to reset this manually.

Note that the z/OS product with CICS uses the message sequence number of the message it last sent, while all other platforms use the sequence number of the next
message to be sent. This means you must reset the message sequence number to 0 at the z/OS (with CICS) end of a channel and to 1 everywhere else.

**Channel configuration**

The following sections detail the configuration to be performed on the z/OS queue manager to implement the channel described in Figure 32 on page 101.

Examples are given for connecting WebSphere MQ for z/OS and MQSeries for OS/2 Warp. If you wish to connect to another WebSphere MQ product use the appropriate set of values from the table in place of those for OS/2.

**Note:** The words in **bold** are user-specified and reflect the names of WebSphere MQ objects used throughout these examples. If you change the names used here, ensure that you also change the other references made to these objects throughout this book. All others are keywords and should be entered as shown.

Table 41. Configuration worksheet for WebSphere MQ for z/OS using CICS

<table>
<thead>
<tr>
<th>ID</th>
<th>Parameter Name</th>
<th>Reference</th>
<th>Example Used</th>
<th>User Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>Queue Manager Name</td>
<td></td>
<td>MVS</td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>Local queue name</td>
<td></td>
<td>MVS.LOCALQ</td>
<td></td>
</tr>
</tbody>
</table>

**Connection to MQSeries for OS/2 Warp**

The values in this section of the table must match those used in Table 16 on page 161 as indicated.

| C  | Remote queue manager name      | A         | OS2               |            |
| D  | Remote queue name              |          | OS2.REMOTEQ      |            |
| E  | Queue name at remote system    | B         | OS2.LOCALQ       |            |
| F  | Transmission queue name        |          | OS2               |            |
| K  | Sender (LU 6.2 using CICS) channel name |   | MVS.OS2.CICS    |            |
| L  | Receiver (LU 6.2 using CICS) channel name |  | OS2.MVS.CICS    |            |

**Connection to WebSphere MQ for Windows**

The values in this section of the table must match those used in Table 18 on page 184 as indicated.

| C  | Remote queue manager name      | A         | WINNT             |            |
| D  | Remote queue name              |          | WINNT.REMOTEQ    |            |
| E  | Queue name at remote system    | B         | WINNT.LOCALQ     |            |
| F  | Transmission queue name        |          | WINNT             |            |
| K  | Sender (LU 6.2 using CICS) channel name |   | MVS.WINNT.CICS  |            |
| L  | Receiver (LU 6.2 using CICS) channel name |  | WINNT.MVS.CICS  |            |

**Connection to WebSphere MQ for AIX**

The values in this section of the table must match those used in Table 22 on page 210 as indicated.

| C  | Remote queue manager name      | A         | AIX               |            |
| D  | Remote queue name              |          | AIX.REMOTEQ      |            |
| E  | Queue name at remote system    | B         | AIX.LOCALQ       |            |
| F  | Transmission queue name        |          | AIX               |            |
| K  | Sender (LU 6.2 using CICS) channel name |   | MVS.AIX.CICS    |            |
| L  | Receiver (LU 6.2 using CICS) channel name |  | AIX.MVS.CICS    |            |

**Connection to WebSphere MQ for HP-UX**

The values in this section of the table must match those used in Table 25 on page 239 as indicated.
## z/OS using CICS configuration

### Table 41. Configuration worksheet for WebSphere MQ for z/OS using CICS (continued)

<table>
<thead>
<tr>
<th>ID</th>
<th>Parameter Name</th>
<th>Reference</th>
<th>Example Used</th>
<th>User Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>C</td>
<td>Remote queue manager name</td>
<td></td>
<td>HPUX</td>
<td></td>
</tr>
<tr>
<td>D</td>
<td>Remote queue name</td>
<td></td>
<td>HPUX.REMOTEQ</td>
<td></td>
</tr>
<tr>
<td>E</td>
<td>Queue name at remote system</td>
<td>F</td>
<td>HPUX.LOCALQ</td>
<td></td>
</tr>
<tr>
<td>F</td>
<td>Transmission queue name</td>
<td></td>
<td>HPUX</td>
<td></td>
</tr>
<tr>
<td>K</td>
<td>Sender (LU 6.2 using CICS) channel name</td>
<td></td>
<td>MVS.HPUX.CICS</td>
<td></td>
</tr>
<tr>
<td>L</td>
<td>Receiver (LU 6.2 using CICS) channel name</td>
<td></td>
<td>HPUX.MVS.CICS</td>
<td></td>
</tr>
</tbody>
</table>

**Connection to MQSeries for AT&T GIS UNIX**

The values in this section of the table must match those used in Table 27 on page 253 as indicated.

<table>
<thead>
<tr>
<th>ID</th>
<th>Parameter Name</th>
<th>Reference</th>
<th>Example Used</th>
<th>User Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>C</td>
<td>Remote queue manager name</td>
<td></td>
<td>GIS</td>
<td></td>
</tr>
<tr>
<td>D</td>
<td>Remote queue name</td>
<td></td>
<td>GIS.REMOTEQ</td>
<td></td>
</tr>
<tr>
<td>E</td>
<td>Queue name at remote system</td>
<td>F</td>
<td>GIS.LOCALQ</td>
<td></td>
</tr>
<tr>
<td>F</td>
<td>Transmission queue name</td>
<td></td>
<td>GIS</td>
<td></td>
</tr>
<tr>
<td>K</td>
<td>Sender (LU 6.2 using CICS) channel name</td>
<td></td>
<td>MVS.GIS.CICS</td>
<td></td>
</tr>
<tr>
<td>L</td>
<td>Receiver (LU 6.2 using CICS) channel name</td>
<td></td>
<td>GIS.MVS.CICS</td>
<td></td>
</tr>
</tbody>
</table>

**Connection to WebSphere MQ for iSeries**

The values in this section of the table must match those used in Table 49 on page 535 as indicated.

<table>
<thead>
<tr>
<th>ID</th>
<th>Parameter Name</th>
<th>Reference</th>
<th>Example Used</th>
<th>User Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>C</td>
<td>Remote queue manager name</td>
<td></td>
<td>AS400</td>
<td></td>
</tr>
<tr>
<td>D</td>
<td>Remote queue name</td>
<td></td>
<td>AS400.REMOTEQ</td>
<td></td>
</tr>
<tr>
<td>E</td>
<td>Queue name at remote system</td>
<td>F</td>
<td>AS400.LOCALQ</td>
<td></td>
</tr>
<tr>
<td>F</td>
<td>Transmission queue name</td>
<td></td>
<td>AS400</td>
<td></td>
</tr>
<tr>
<td>K</td>
<td>Sender (LU 6.2 using CICS) channel name</td>
<td></td>
<td>MVS.AS400.CICS</td>
<td></td>
</tr>
<tr>
<td>L</td>
<td>Receiver (LU 6.2 using CICS) channel name</td>
<td></td>
<td>AS400.MVS.CICS</td>
<td></td>
</tr>
</tbody>
</table>

### WebSphere MQ for z/OS sender-channel definitions using CICS

**Local Queue**

Object type : QLOCAL  
Name : OS2  
Usage : X (XmitQ)

**Remote Queue**

Object type : QREMOTE  
Name : OS2.REMOTEQ  
Name on remote system : OS2.LOCALQ  
Remote system name : OS2  
Transmission queue : OS2

**Sender Channel**

Channel name : MVS.OS2.CICS  
Channel type : 1 (Sender)  
Target system id : <blank>  
Transmission queue name : OS2  
Transaction id : CKSG  
Connection name : OS2  
LU62 TP name : MQSERIES

### WebSphere MQ for z/OS receiver-channel definitions using CICS

**Local Queue**

Object type : QLOCAL  
Name : MVS.LOCALQ  
Usage : N (Normal)
Receiver Channel
Channel name: **OS2.MVS.CICS**
Channel type: 3 (Receiver)
Target system id: <blank>
z/OS using CICS configuration
Chapter 32. Message channel planning example for z/OS using CICS

This chapter provides a detailed example of how to connect queue managers together to send messages from one to the other. The example gives you a step-by-step implementation of a unidirectional interconnection of two queue managers.

Figure 79 illustrates the interaction between all the system components used for transferring messages between queue managers.

In the following list, the numbered items refer to the boxed index numbers in the figure.

1. The "Payroll reporter" application connects to queue manager "QM1", opens a queue called "Payrollr", and places messages on the queue.

2. The attributes of Payrollr in queue manager QM1 are:

   - QUEUE Payrollr
   - TYPE QREMOTE
   - DESCR PAYROLL QUEUE ON QM2 QUEUE MANAGER
   - PUT ENABLED
   - DEFPRTY 0
   - DEFPSIST YES
   - RNAME QM1_payroll
   - RQMNAME QM2

From this information, the local queue manager QM1 determines that messages for this queue have to be transmitted to a remote queue manager QM2.
Planning example for z/OS using CICS

For QM1, QM2 is just a transmission queue on which messages have to be placed. A transmission queue is a local queue with its usage parameter set to XMITQ.

3. The attributes of the transmission queue, QM2, in queue manager QM1 are:

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>QUEUE</td>
<td>QM2</td>
</tr>
<tr>
<td>TYPE</td>
<td>LOCAL</td>
</tr>
<tr>
<td>DESCRIPTOR</td>
<td>QUEUE MANAGER QM2 TRANSMISSION QUEUE</td>
</tr>
<tr>
<td>PUT</td>
<td>ENABLED</td>
</tr>
<tr>
<td>DEFPRTY</td>
<td>0</td>
</tr>
<tr>
<td>DEFPSIST</td>
<td>YES</td>
</tr>
<tr>
<td>PROCO</td>
<td>0</td>
</tr>
<tr>
<td>IPROCS</td>
<td>0</td>
</tr>
<tr>
<td>CURDEPTH</td>
<td>0</td>
</tr>
<tr>
<td>MAXDEPTH</td>
<td>100000</td>
</tr>
<tr>
<td>PROCESS</td>
<td>QM2.PROCESS</td>
</tr>
<tr>
<td>TRIGGER</td>
<td></td>
</tr>
<tr>
<td>MAXMSG</td>
<td>4194304</td>
</tr>
<tr>
<td>BOTHRESH</td>
<td>0</td>
</tr>
<tr>
<td>BOQNAME</td>
<td></td>
</tr>
<tr>
<td>STGCLASS</td>
<td>DEFAULT</td>
</tr>
<tr>
<td>INITQ</td>
<td>Init_queue</td>
</tr>
<tr>
<td>USAGE</td>
<td>XMITQ</td>
</tr>
<tr>
<td>SHARE</td>
<td></td>
</tr>
<tr>
<td>DEFSOFT</td>
<td>EXCL</td>
</tr>
<tr>
<td>MSGDLVSQ</td>
<td>FIFO</td>
</tr>
<tr>
<td>RETINTVL</td>
<td>0</td>
</tr>
<tr>
<td>TRIGTYPE</td>
<td>FIRST</td>
</tr>
<tr>
<td>TRIGDEPTH</td>
<td>1</td>
</tr>
<tr>
<td>TRIGMPRI</td>
<td>0</td>
</tr>
<tr>
<td>TRIGGERDATA</td>
<td>0</td>
</tr>
<tr>
<td>DEFTYPE</td>
<td>PREDEFINED</td>
</tr>
<tr>
<td>NOHARDENBO</td>
<td></td>
</tr>
<tr>
<td>GET</td>
<td>ENABLED</td>
</tr>
</tbody>
</table>

Messages that the application puts to Payrollr are actually placed on the transmission queue QM2.

4. In this example, assume that the payroll message is the first message to be placed on the empty transmission queue, and because of the triggering attributes of the transmission queue, the queue manager determines that a trigger message is to be issued.

The transmission queue definition refers to an initiation queue called Init_queue, and the queue manager places a trigger message on this queue. The transmission queue definition also refers to the trigger process definition, and information from this definition is included in the trigger message.
The definition of the process in queue manager QM1 is:

<table>
<thead>
<tr>
<th>PROCESS</th>
<th>QM2.PROCESS</th>
</tr>
</thead>
<tbody>
<tr>
<td>DESCR</td>
<td>PROCESS DEFINITION - TO TRIGGER CHANNEL QM1.2.QM2.CHANNEL</td>
</tr>
<tr>
<td>APPLTYPE</td>
<td>CICS</td>
</tr>
<tr>
<td>APPLICID</td>
<td>CKSG</td>
</tr>
<tr>
<td>USERDATA</td>
<td>QM1.2.QM2.CHANNEL</td>
</tr>
<tr>
<td>ENVRDATA</td>
<td>environment information</td>
</tr>
</tbody>
</table>

The result of this trigger processing is that a trigger message is placed on the initiation queue, Init_queue.

5. If you experience trigger messages failing to appear when expected, refer to the [WebSphere MQ Application Programming Guide](#).

6. The CKTI transaction is a long-running task that monitors the initiation queue, Init_queue. CKTI processes the trigger message, an MQTM structure, to find that it must start CKSG. CKSG is the CICS name of the sender channel MCA transaction.

7. CKTI starts CKSG, passing the MQTM structure. The CKSG transaction starts processing, receives the MQTM structure, and extracts the name of the channel.

8. The channel name is used by CKSG to get the channel definition from the channel definition file on QM1. The DQM display settings panel of the channel in QM1.2.QM2.CHANNEL, is:
Planning example for z/OS using CICS

Figure 80. Sender settings (1)
The channel definition shows that CKSG must allocate a session on the CICS QM2C connection and invoke the CKRC transaction at the destination CICS system.

9. The QM2C connection definition provides a communications link to the CICS system at the remote installation. The definition is as follows:

**Figure 81. Sender settings (2)**

The channel definition shows that CKSG must allocate a session on the CICS QM2C connection and invoke the CKRC transaction at the destination CICS system.

9. The QM2C connection definition provides a communications link to the CICS system at the remote installation. The definition is as follows:

**Figure 82. Connection definition (1)**

Planning example for z/OS using CICS
10. The connection definition on the remote installation CICS system is called QM1C, and is defined as follows:

**Figure 83. Connection definition (2)**

**Figure 84. Connection definition (1)**
11. CKRC is started by CICS on the remote system, and is passed the channel name during the initial data flows.

12. The transaction CKRC reads the definition for the receiver channel QM1.2.QM2.CHANNEL from the channel definition file, which contains:

```
OBJECT CHARACTERISTICS
CEDA View
+ INService : Yes Yes | No
SECURITY
SECURITYname :
ATTachsec : Local Local | Identify | Verify | Persistent
| Mixidpe
BINDPassword :
BINDSecurity : No No | Yes
```

```
APPLID=CICSTQMI
PF 1 HELP 3 END 6 CRSR 7 SBH 8 SFH 9 MSG 10 SB 11 SF 12 CNCL
```

Figure 85. Connection definition (2)

Chapter 32. Message channel planning example for z/OS using CICS
13. Once the message channel has completed the startup negotiation, the sender channel passes messages to the receiver channel. The receiver channel takes the name of the queue manager, queue name and message descriptor from the transmission header, and issues an MQPUT1 call to put the message on the local queue, QM1_payroll.

When the batch limit of 100 is reached, or when the transmission queue is empty, the sender and receiver channels issue a syncpoint to commit the changes through the queue managers.

14. The commit action by the QM2 queue manager makes the messages available to the “Payroll process” application.
Chapter 33. Preparing WebSphere MQ for z/OS for DQM with queue-sharing groups

This chapter describes the concept of distributed queuing with queue-sharing groups on WebSphere MQ for z/OS.

It also describes the components of distributed queuing in this environment and the WebSphere MQ preparations required before DQM can be used with queue-sharing groups, and the advantages of using DQM with queue-sharing groups.

For information on how to monitor and control channels when using queue-sharing groups, see Chapter 23, “Monitoring and controlling channels on z/OS” on page 341.

Concepts

This section describes the concepts related to distributed queuing with queue-sharing groups. For additional information on the concepts of shared queues and queue-sharing groups, see WebSphere MQ for z/OS Concepts and Planning Guide, “Shared queues”.

Class of service

A shared queue is a type of local queue that offers a different class of service. Messages on a shared queue are stored in a coupling facility (CF), which allows them to be accessed by all queue managers in the queue-sharing group. A message on a shared queue must be a message of length no more than 63KB.

Generic interface

A queue-sharing group has a generic interface that allows the network to view the group as a single entity. This is achieved by having a single generic address that can be used to connect to any queue manager within the group.

Each queue manager in the queue-sharing group listens for inbound session requests on an address that is logically related to the generic address. For more information see “Listeners” below.

Components

What follows is a description of the components required to enable distributed queuing with queue-sharing groups.

Listeners

The group LU 6.2 and TCP/IP listeners listen on an address that is logically connected to the generic address.

For the LU 6.2 listener, the specified LUGROUP is mapped to the VTAM generic resource associated with the queue-sharing group. For an example of setting up this technology, see Table 32 on page 363.
For the TCP/IP listener, the specified port has two mutually exclusive means of being connected to the generic address:

- In the case of a front-end router such as the IBM Network dispatcher (see *Network Dispatcher User’s Guide, GC31–8496–03*), inbound connect requests are forwarded from the router to the members of the queue-sharing group.

- In the case of TCP/IP WLM/DNS, each listener registers as being part of the WLM group. This is a registration type model, similar to the VTAM generic resource for LU 6.2. For an example of setting up this technology, see “Using WLM/DNS” on page 453. WLM/DNS only maps hostname and does not map port numbers. This means that all the group listeners in a queue-sharing group must use the same port number. Use the WebSphere MQ command (MQSC) as shown in the following examples:

  - On queue manager QM1:
    
    ```bash
    START LSTR PORT(2424) INDISP(GROUP) + IPADDR(QM1.MACH.IBM.COM)
    ```

  - On queue manager QM2:
    
    ```bash
    START LSTR PORT(2424) INDISP(GROUP) + IPADDR(QM2.MACH.IBM.COM)
    ```

**Transmission queues and triggering**

A shared transmission queue is used to store messages before they are moved from the queue-sharing group to the destination. It is a shared queue and it is accessible to all queue managers in the queue-sharing group.

**Triggering**

A triggered shared queue can generate more than one trigger message for a satisfied trigger condition. There is one trigger message generated for each local initiation queue defined on a queue manager in the queue-sharing group associated with the triggered shared queue.

In the case of distributed queuing, each channel initiator receives a trigger message for a satisfied shared transmission queue trigger condition. However, only one channel initiator will actually process the triggered start, and the others will fail safely. The triggered channel is then started with a load balanced start (see “Load-balanced channel start” on page 441) that will be triggered to start channel QSG.TO.QM2. To create a shared transmission queue, use the WebSphere MQ commands (MQSC) as shown in the following example:

```bash
DEFINE QLOCAL(QM2) DESCR('Transmission queue to QM2') + USAGE(XMITQ) QSGDISP(SHARED) + CFSTRUCT(APPLICATION1) INITQ(SYSTEM.CHANNEL.INITQ) + TRIGGER TRIGDATA(QSG.TO.QM2)
```

**Message channel agents**

A channel can only be started on a channel initiator if it has access to a channel definition for a channel with that name. A channel definition can be defined to be private to a queue manager or stored on the shared repository and available anywhere (a group definition). This means that a group defined channel is available on any channel initiator in the queue-sharing group.

**Note:** The private copy of the group definition can be changed or deleted.

To create group channel definitions, use the WebSphere MQ commands (MQSC) as shown in the following examples:
There are two perspectives from which to look at the message channel agents used for distributed queuing with queue-sharing groups:

- Inbound
- Outbound

**Inbound**
An inbound channel is a shared channel if it is connected to the queue manager through the group listener. It is connected either through the generic interface to the queue-sharing group, then directed to a queue manager within the group, or targeted at a specific queue manager’s group port or the luname used by the group listener.

**Outbound**
An outbound channel is a shared channel if it moves messages from a shared transmission queue. In the above example commands, sender channel QSG.TO.QM2 is a shared channel because its transmission queue, QM2 is defined with QSGDISP(SHARED).

**Synchronization queue**
Shared channels have their own shared synchronization queue called SYSTEM.QSG.CHANNEL.SYNCQ, which is accessible to any member of the queue-sharing group. (Private channels continue to use the private synchronization queue. See “Synchronization queue” on page 358) This means that the channel can be restarted on a different queue manager and channel initiator instance within the queue-sharing group in the event of failure of the communications subsystem, channel initiator or queue manager. (See “Shared channel recovery” on page 442 for details.)

DQM with queue-sharing groups requires that a shared queue is available with the name SYSTEM.QSG.CHANNEL.SYNCQ. This queue must be available so that a group listener can successfully start.

If a group listener fails because the queue was not available, the queue can be defined and the listener can be restarted without recycling the channel initiator, and the non-shared channels are not affected.

Make sure that you define this queue using INDXTYPE(MSGID). This will improve the speed at which they can be accessed.

**Benefits**

The following section describes the benefits of shared queuing, which are:

- Load-balanced channel start
- Shared channel recovery
- Client channels

**Load-balanced channel start**
A shared transmission queue can be serviced by an outbound channel running on any channel initiator in the queue-sharing group. Load-balanced channel start
determines where a start channel command is targeted. An appropriate channel initiator is chosen that has access to the necessary communications subsystem. For example, a channel defined with TRPTYPE(LU6.2) will not be started on a channel initiator that only has access to a TCP/IP subsystem.

The choice of channel initiator is dependant on the channel load and the headroom of the channel initiator. The channel load is the number of active channels as a percentage of the maximum number of active channels allowed as defined in the channel initiator parameters. The headroom is the difference between the number of active channels and the maximum number allowed.

Inbound shared channels can be load-balanced across the queue-sharing group by use of a generic address, as described in “Listeners” on page 439.

### Shared channel recovery

The following table shows the types of shared-channel failure and how each type is handled.

<table>
<thead>
<tr>
<th>Type of failure:</th>
<th>What happens:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Channel initiator communications subsystem failure</td>
<td>The channels dependent on the communications subsystem enter channel retry, and are restarted on an appropriate queue-sharing group channel initiator by a load-balanced start command.</td>
</tr>
<tr>
<td>Channel initiator failure</td>
<td>The channel initiator fails, but the associated queue manager remains active. The queue manager monitors the failure and initiates recovery processing.</td>
</tr>
<tr>
<td>Queue manager failure</td>
<td>The queue manager fails (failing the associated channel initiator). Other queue managers in the queue-sharing group monitor the event and initiate peer recovery.</td>
</tr>
<tr>
<td>Shared status failure</td>
<td>Channel state information is stored in DB2, so a loss of connectivity to DB2 becomes a failure when a channel state change occurs. Running channels can carry on running without access to these resources. On a failed access to DB2, the channel enters retry.</td>
</tr>
</tbody>
</table>

### Client channels

Client connection channels can benefit from the high availability of messages in queue-sharing groups that are connected to the generic interface instead of being connected to a specific queue manager. (For more information about this, see the WebSphere MQ Clients manual.)

### Clusters and queue-sharing groups

You can make your shared queue available to a cluster in a single definition. To do this you specify the name of the cluster when you define the shared queue.

Users in the network see the shared queue as being hosted by each queue manager within the queue-sharing group (the shared queue is not advertised as being hosted by the queue-sharing group). Clients can start sessions with all members of the queue-sharing group to put messages to the same shared queue.

For more information, see WebSphere MQ Queue Manager Clusters.
Channels and serialization

If a queue manager in a queue-sharing group fails while a message channel agent is dealing with uncommitted messages on one or more shared queues, the channel and the associated channel initiator will end, and shared queue peer recovery will take place for the queue manager.

Because shared queue peer recovery is an asynchronous activity, peer channel recovery might try to simultaneously restart the channel in another part of the queue sharing group before shared queue peer recovery is complete. If this happens, committed messages might be processed ahead of the messages still being recovered. To ensure that messages are not processed out of sequence in this way, message channel agents that process messages on shared queues serialize their access to these queues by issuing the MQCONNX API call.

An attempt to start a channel for which shared queue peer recovery is still in progress might result in a failure. An error message indicating that recovery is in progress is issued, and the channel is put into retry state. Once queue manager peer recovery is complete, the channel can restart at the time of the next retry.

An attempt to RESOLVE, PING, or DELETE a channel can fail for the same reason.

Note: Serialized access to shared queues is not supported for MQI channels.

Intra-group queuing

Intra-group queuing (IGQ) can effect potentially fast and less-expensive small message transfer between queue managers within a queue-sharing group (QSG), without the need to define channels between the queue managers. That is, intra-group queuing can be used to deliver, more efficiently, small messages to queues residing on remote queue managers within a queue-sharing group. See Chapter 37, “Intra-group queuing” on page 461 for more information.
Chapter 34. Setting up communication for WebSphere MQ for z/OS using queue-sharing groups

When a distributed-queuing management channel is started, it tries to use the connection specified in the channel definition. For this to succeed, it is necessary for the connection to be defined and available. This section explains how to do this.

You might also find it useful to refer to Chapter 35, “Example configuration - WebSphere MQ for z/OS using queue-sharing groups” on page 447.

Deciding on a connection

There are two forms of communication protocol that can be used:
• TCP
• LU 6.2 through APPC/MVS

Defining a TCP connection

For information on setting up your TCP, see “Defining a TCP connection” on page 361.

Sending end

The connection name (CONNAME) field in the channel definition to connect to your queue sharing group should be set to the generic interface of your queue-sharing group (see “Generic interface” on page 439). If you are using DNS/WLM, the generic interface is the name in DNSGROUP in your channel initiator parameters. If it is not set, it is the queue-sharing group name. For details of DNSGROUP, see WebSphere MQ for z/OS System Setup Guide.

Receiving on TCP using a queue-sharing group

Receiving shared channel programs are started in response to a startup request from the sending channel. To do this, a listener has to be started to detect incoming network requests and start the associated channel. You start this listener program with the START LISTENER command, using the inbound disposition of the group, or using the operations and control panels.

All group listeners in the queue-sharing group must be listening on the same port. If you have more than one channel initiator running on a single MVS image you can define virtual IP addresses and start your TCP listener program to only listen on a specific address or hostname by specifying IPADDR in the START LISTENER command. (For more information, see Chapter 33, “Preparing WebSphere MQ for z/OS for DQM with queue-sharing groups” on page 439.)

Defining an LU6.2 connection

For information on setting up APPC/MVS, see Chapter 25, “Setting up communication for z/OS” on page 361.
Connecting to APPC/MVS (LU 6.2)

The connection name (CONNAME) field in the channel definition to connect to your queue-sharing group should be set to the symbolic destination name, as specified in the side information data set for APPC/MVS. The partner LU specified in this symbolic destination should be the generic resource name. See “Defining yourself to the network using generic resources” on page 450 for more details.

Receiving on LU 6.2 using a generic interface

Receiving shared MCAs are started in response to a startup request from the sending channel. To do this, a group listener program has to be started to detect incoming network requests and start the associated channel. The listener program is an APPC/MVS server. You start it with the START LISTENER command, using an inbound disposition group, or using the operations and control panels. You must specify the LU name to use by means of a symbolic destination name defined in the side information data set. See “Defining yourself to the network using generic resources” on page 450 for more details.
Chapter 35. Example configuration - WebSphere MQ for z/OS using queue-sharing groups

This chapter gives an example of how to set up communication links into a queue-sharing group on WebSphere MQ for z/OS or MQSeries for OS/390 to WebSphere MQ products on the following platforms:
- OS/2
- Windows
- AIX

(You can also connect any of the following:
  - z/OS to z/OS
  - z/OS to MVS/ESA

Setting up communication links from a queue-sharing group to a distributed platform is the same as described in Chapter 26, “Example configuration - IBM WebSphere MQ for z/OS” on page 363. There are examples to other platforms in that chapter.

First it describes the parameters needed for an LU 6.2 connection; then it describes:
- “Establishing an LU 6.2 connection into a queue-sharing group” on page 450
- “Establishing a TCP connection into a queue-sharing group” on page 453

When the connection is established, you need to define some channels to complete the configuration. This is described in “WebSphere MQ for z/OS shared channel configuration” on page 454.

See Chapter 7, “Example configuration chapters in this book” on page 101 for background information about this chapter and how to use it.

Configuration parameters for an LU 6.2 connection

Table 42 on page 448 presents a worksheet listing all the parameters needed to set up communication from z/OS to one of the other WebSphere MQ platforms. The worksheet shows examples of the parameters, which have been tested in a working environment, and leaves space for you to fill in your own values. An explanation of the parameter names follows the worksheet. Use the worksheet in this chapter in conjunction with the worksheet in the chapter for the platform to which you are connecting.
z/OS and LU 6.2

The steps required to set up an LU 6.2 connection are described in “Establishing an LU 6.2 connection into a queue-sharing group” on page 450 with numbered cross references to the parameters on the worksheet.

**Configuration worksheet**

Use this worksheet to record the values you use for your configuration. Where numbers appear in the Reference column they indicate that the value must match that in the appropriate worksheet elsewhere in this book. The examples that follow in this chapter refer back to the values in the ID column. The entries in the Parameter Name column are explained in “Explanation of terms” on page 449.

<table>
<thead>
<tr>
<th>ID</th>
<th>Parameter Name</th>
<th>Reference</th>
<th>Example Used</th>
<th>User Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Command prefix</td>
<td>+CPF</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Network ID</td>
<td>NETID</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Node name</td>
<td>MVSPU</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>Modename</td>
<td>#INTER</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>Local Transaction Program name</td>
<td>MQSERIES</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>LAN destination address</td>
<td>400074511092</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>Local LU name</td>
<td>MVSLU1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>Generic resource name</td>
<td>MVSGR</td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>Symbolic destination</td>
<td>G1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>Symbolic destination for generic resource name</td>
<td>G2</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Connection to an OS/2 system**

The values in this section of the table must match those used in Table 15 on page 144 as indicated.

<table>
<thead>
<tr>
<th>ID</th>
<th>Parameter Name</th>
<th>Reference</th>
<th>Example Used</th>
<th>User Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>13</td>
<td>Symbolic destination</td>
<td></td>
<td>M2</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>Modename</td>
<td>#INTER</td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>Remote Transaction Program name</td>
<td>MQSERIES</td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>Partner LU name</td>
<td>OS2LU</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Connection to a Windows system**

The values in this section of the table must match those used in Table 17 on page 168 as indicated.

<table>
<thead>
<tr>
<th>ID</th>
<th>Parameter Name</th>
<th>Reference</th>
<th>Example Used</th>
<th>User Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>13</td>
<td>Symbolic destination</td>
<td>M3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>Modename</td>
<td>#INTER</td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>Remote Transaction Program name</td>
<td>MQSERIES</td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>Partner LU name</td>
<td>WINNTLU</td>
<td></td>
<td></td>
</tr>
<tr>
<td>21</td>
<td>Remote node ID</td>
<td>05D 30F65</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Connection to an AIX system**

The values in this section of the table must match those used in Table 21 on page 197 as indicated.

<table>
<thead>
<tr>
<th>ID</th>
<th>Parameter Name</th>
<th>Reference</th>
<th>Example Used</th>
<th>User Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>13</td>
<td>Symbolic Destination</td>
<td>M4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>Modename</td>
<td>#INTER</td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>Remote Transaction Program name</td>
<td>MQSERIES</td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>Partner LU name</td>
<td>AIXLU</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Explanation of terms

1 Command prefix
   This is the unique command prefix of your WebSphere MQ for z/OS queue-manager subsystem. The z/OS systems programmer defines this at installation time, in SYS1.PARMLIB(IEFSSNss), and will be able to tell you the value.

2 Network ID
   The VTAM startup procedure in your installation is partly customized by the ATCSTRxx member of the data set referenced by the DDNAME VTAMLST. The Network ID is the value specified for the NETID parameter in this member. For Network ID you must specify the name of the NETID that owns the WebSphere MQ communications subsystem (WebSphere MQ channel initiator or CICS for z/OS as the case may be). Your network administrator will tell you the value.

3 Node name
   VTAM, being a low-entry network node, does not have a Control Point name for Advanced Peer-to-Peer Networking (APPN) use. It does however have a system services control point name (SSCPNAME). For node name, you must specify the name of the SSCP that owns the WebSphere MQ communications subsystem (WebSphere MQ channel initiator or CICS for z/OS as the case may be). This is defined in the same ATCSTRxx member as the Network ID. Your network administrator will tell you the value.

9 Local LU name
   A logical unit (LU) is software that serves as an interface or translator between a transaction program and the network. It manages the exchange of data between transaction programs. The local LU name is the unique VTAM APPLID of this WebSphere MQ subsystem. Your network administrator will tell you this value.

11 12 13 Symbolic destination
   This is the name you give to the CPI-C side information profile. You need a side information entry for each LU 6.2 listener.

14 Modename
   This is the name given to the set of parameters that control the LU 6.2 conversation. An entry with this name and similar attributes must be defined at each end of the session. In VTAM, this corresponds to a mode table entry. You network administrator will assign this to you.

7 15 Transaction Program name
   WebSphere MQ applications trying to converse with this queue manager will specify a symbolic name for the program to be run at the receiving end. This will have been specified in the TPNAME attribute on the channel definition at the sender. For simplicity, wherever possible use a transaction program name of MQSERIES, or in the case of a connection to VSE/ESA, where the length is limited to 4 bytes, use MQTP.

   See Table 32 on page 363 for more information. If the receiving end is z/OS using CICS, special values are required.

8 LAN destination address
   This is the LAN destination address that your partner nodes will use to communicate with this host. When you are using a 3745 network controller, it will be the value specified in the LOCADD parameter for the line definition to which your partner is physically connected. If your
partner nodes use other devices such as 317X or 6611 devices, the address will have been set during the customization of those devices. Your network administrator will tell you this value.

10 Generic resource name
A generic resource name is a unique name assigned to a group of LU names used by the channel initiators in a queue-sharing group.

16 Partner LU name
This is the LU name of the WebSphere MQ queue manager on the system with which you are setting up communication. This value is specified in the side information entry for the remote partner.

21 Remote node ID
For a connection to Windows NT, this is the ID of the local node on the Windows NT system with which you are setting up communication.

Establishing an LU 6.2 connection into a queue-sharing group

To establish and LU 6.2 connection, there are two steps:
1. Define yourself to the network using generic resources.
2. Define a connection to the partner.

Defining yourself to the network using generic resources
This example describes how to use VTAM Generic Resources to have one connection name to connect to the queue-sharing group.

1. SYS1.PARMLIB(APPCPMMxx) contains the start-up parameters for APPC. You must add a line to this file to define the local LU name you intend to use for the WebSphere MQ LU 6.2 group listener. The line you add should take the form

   LUADD ACBNAME(mvslu1)
   NOSCHED
   TPDATA(csq.appctp)
   GRNAME(mvsgr)

Specify values for ACBNAME (9), TPDATA and GRNAME (10).

The NOSCHED parameter tells APPC that our new LU will not be using the LU 6.2 scheduler (ASCH), but has one of its own. TPDATA refers to the Transaction Program data set in which LU 6.2 stores information about transaction programs. Again, WebSphere MQ will not use this, but it is required by the syntax of the LUADD command.

2. Start the APPC subsystem with the command:

   START APPC, SUB=MSTR, APPC=xx

   where xx is the suffix of the PARMLIB member in which you added the LU in step 1.

   Note: If APPC is already running, it can be refreshed with the command:

   SET APPC=xx

   The effect of this is cumulative, that is, APPC will not lose its knowledge of objects already defined to it in this or another PARMLIB member.

3. Add the new LU to a suitable VTAM major node definition. These are typically in SYS1.VTAMLST. The APPL definition will look similar to the sample shown.
4. Activate the major node. This can be done with the command:

\`V,NET,ACT,majornode\`

5. Add entries defining your LU and generic resource name to the CPI-C side information data set. Use the APPC utility program ATBSDFMU to do this. Sample JCL is in `thlqual.SCSQPROC(CSQ4SIDE)` (where `thlqual` is the target library high-level qualifier for WebSphere MQ data sets in your installation.) The entries you add will look like this:

```
SIADD
   DESTNAME(G1)
   MODENAME(#INTER)
   TPNAME(MQSERIES)
   PARTNER_LU(MVSLU1)
SIADD
   DESTNAME(G2)
   MODENAME(#INTER)
   TPNAME(MQSERIES)
   PARTNER_LU(MVSGR)
```

6. Create the channel-initiator parameter module for your queue manager. Sample JCL to do this is in `thlqual.SCSQPROC(CSQ4XPRM)`. You must specify the local LU ( ) assigned to your queue manager in the LUGROUP= parameter of the CSQ6CHIP macro.
LU 6.2 without CICS

```
//SYSIN  DD *
CSQ6CHIP ADAPS=8,                         X
  ACTCHL=200,                             X
  ADOPTMCA=NO,                            X
  ADOPTCHK=ALL,                           X
  CURRCHL=200,                            X
  DISPS=5,                                X
  DNSGROUP=,                              X
  DNSWLM=NO,                              X
  LSTRTM=60,                              X
  LUGROUP=MVSLU1,                         X
  LUNAME=MVSLU,                           X
  LU62ARM=,                               X
  LU62CHL=200,                            X
  OPORTMIN=0,                             X
  OPORTMAX=0,                             X
  TCPCHL=200,                             X
  TCPKEEP=NO,                             X
  TCPNAME=TCPIP,                          X
  TCPTYPE=OESOCKET,                       X
  TRAXSTR=YES,                            X
  TRAXTBL=2,                              X
  SERVICE=0
END
/*
```

Figure 88. Channel Initiator initialization parameters

7. Modify the job to assemble and link-edit the tailored version of the initiator macro to produce a new load module.
8. Submit the job and verify that it completes successfully.
9. Put the new initialization-parameters module in an APF-authorized user library. Include this library in the STEPLIB concatenation for the channel initiator’s started-task procedure, ensuring that it precedes the library `thlqual.SCSQAUTH`.

Defining a connection to a partner

Note: This example is for a connection to an OS/2 system but the task is the same for other platforms.

Add an entry to the CPI-C side information data set to define the connection. Sample JCL to do this is in `thlqual.SCSQPROC(CSQ4SIDE)`. The entry you add will look like this:

```
SIADD
  DESTNAME(M2) 13
  MODENAME(#INTER) 14
  TPNAME(MQSERIES) 15
  PARTNER_LU(OS2LU) 16
```

What next?

The connection is now established. You are ready to complete the configuration. Go to “WebSphere MQ for z/OS shared channel configuration” on page 454.
Establishing a TCP connection into a queue-sharing group

Edit the channel initiator initialization parameters. Sample JCL to do this is in thlqual.SCSQPROC(CSQ4XPRM). You must add the name of the TCP address space to the TCPNAME= parameter.

Using WLM/DNS

Edit the channel initiator initialization parameters. You must set DNSWLM=YES; optionally you can add the name of the group name to be used as a hostname to the DNSGROUP= parameter. If you leave the name blank the queue-sharing group name is used.

```
//SYSIN DD *
CSQ6CHIP ADAPS=8, X
ACTCHL=200, X
ADOPTMAC=NO, X
ADOPTCHK=ALL, X
CURRCHL=200, X
DISPS=5, X
DNSGROUP=MYGROUP, X
DNSWLM=YES, X
LSTRTMR=60, X
LUGROUP=, X
LUNAME=MWSLU, X
LU62ARM=, X
LU62CHL=200, X
OPORTMIN=0, X
OPORTMAX=0, X
TCPCHL=200, X
TCPKEEP=NO, X
TCPNAME=TCP1P, X
TCP2TYPE=OESOCKET, X
TRAXSTR=YES, X
TRAXTBL=2, X
SERVICE=0
END
/*

Figure 89. Channel Initiator initialization parameters

WLM/DNS does not offer any support for mapping one incoming port number to a different outgoing port number. This means that each channel initiator must use a different hostname, by one of the following methods:

- Run each channel initiator on a different MVS image
- Run each channel initiator with a different TCP stack on the same MVS image.
- Have multiple Virtual IP Addresses (VIPAs) on one TCP stack.

See z/OS Communications server IP User’s Guide, SC31-8780 for more information on VIPA.

See "Redbook: TCP/IP in a sysplex", SG24-5235 for more information on WLM/DNS.

What next?

The TCP connection is now established. You are ready to complete the configuration. Go to "WebSphere MQ for z/OS shared channel configuration" on page 454.
WebSphere MQ for z/OS shared channel configuration

1. Start the channel initiator using the command:
   
   ```
   +cpf START CHINIT PARM(xparms)
   ```
   
   where *xparms* is the name of the channel-initiator parameter module that you created.

2. Start an LU6.2 group listener using the command:
   
   ```
   +cpf START LSTR LUNAME(G1) TRPTYPE(LU62) INDISP(GROUP)
   ```
   
   The LUNAME of G1 refers to the symbolic name you gave your LU (11).

3. If you are using Virtual IP Addressing and wish to listen on a specific address, use the command:
   
   ```
   +cpf START LSTR PORT(1555) INDISP(GROUP) IPADDR(msvipa)
   ```

WebSphere MQ channels will not initialize successfully if the channel negotiation detects that the message sequence number is different at each end. You may need to reset this manually.

Shared channel configuration

The following sections detail the configuration to be performed on the z/OS queue manager to implement the channel described in Figure 32 on page 101.

Examples are given for connecting WebSphere MQ for z/OS and MQSeries for OS/2 Warp. If you wish to connect to another WebSphere MQ product use the appropriate set of values from the table in place of those for OS/2.

**Note:** The words in **bold** are user-specified and reflect the names of WebSphere MQ objects used throughout these examples. If you change the names used here, ensure that you also change the other references made to these objects throughout this book. All others are keywords and should be entered as shown.

### Table 43. Configuration worksheet for WebSphere MQ for z/OS using queue-sharing groups

<table>
<thead>
<tr>
<th>ID</th>
<th>Parameter Name</th>
<th>Reference</th>
<th>Example Used</th>
<th>User Value</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><strong>Definition for local node</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>Queue Manager Name</td>
<td>QSG</td>
<td>QSG.SHAREDQ</td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>Local queue name</td>
<td>QSG.SHAREDQ</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Connection to MQSeries for OS/2 Warp</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>Remote queue manager name</td>
<td>A</td>
<td>OS2</td>
<td></td>
</tr>
<tr>
<td>D</td>
<td>Remote queue name</td>
<td>OS2.REMOTEQ</td>
<td></td>
<td></td>
</tr>
<tr>
<td>E</td>
<td>Queue name at remote system</td>
<td>B</td>
<td>OS2.LOCALQ</td>
<td></td>
</tr>
<tr>
<td>F</td>
<td>Transmission queue name</td>
<td>OS2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>G</td>
<td>Sender (LU 6.2) channel name</td>
<td>QSG.OS2.SNA</td>
<td>QSG.OS2.SNA</td>
<td></td>
</tr>
<tr>
<td>H</td>
<td>Sender (TCP) channel name</td>
<td>QSG.OS2.TCP</td>
<td></td>
<td></td>
</tr>
<tr>
<td>I</td>
<td>Receiver (LU 6.2) channel name</td>
<td>C</td>
<td>OS2.QSG.SNA</td>
<td></td>
</tr>
<tr>
<td>J</td>
<td>Receiver (TCP) channel name</td>
<td>D</td>
<td>OS2.QSG.TCP</td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Connection to WebSphere MQ for Windows</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>Remote queue manager name</td>
<td>A</td>
<td>WINNT</td>
<td></td>
</tr>
</tbody>
</table>
Table 43. Configuration worksheet for WebSphere MQ for z/OS using queue-sharing groups (continued)

<table>
<thead>
<tr>
<th>ID</th>
<th>Parameter Name</th>
<th>Reference</th>
<th>Example Used</th>
<th>User Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Remote queue name</td>
<td>WINNT.REMOTEQ</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>Queue name at remote system</td>
<td>WINNT.LOCALQ</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Transmission queue name</td>
<td>WINNT</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Sender (LU 6.2) channel name</td>
<td>QSG.WINNT.SNA</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Receiver (LU 6.2) channel name</td>
<td>WINNT.QSG.SNA</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Receiver (TCP/IP) channel name</td>
<td>WINNT.QSG.TCP</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Connection to WebSphere MQ for AIX

The values in this section of the table must match those used in Table 22 on page 210 as indicated.

<table>
<thead>
<tr>
<th>ID</th>
<th>Parameter Name</th>
<th>Reference</th>
<th>Example Used</th>
<th>User Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Remote queue manager name</td>
<td>AIX</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>Remote queue name</td>
<td>AIX.REMOTEQ</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Queue name at remote system</td>
<td>AIX.LOCALQ</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Transmission queue name</td>
<td>AIX</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Sender (LU 6.2) channel name</td>
<td>QSG.AIX.SNA</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Sender (TCP/IP) channel name</td>
<td>QSG.AIX.TCP</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>Receiver (LU 6.2) channel name</td>
<td>AIX.QSG.SNA</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>Receiver (TCP/IP) channel name</td>
<td>AIX.QSG.TCP</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

WebSphere MQ for z/OS shared sender-channel definitions using LU 6.2

Local Queue
- Object type : QLOCAL
- Name : OS2
- Usage : X (XmitQ)
- Disposition : SHARED

Remote Queue
- Object type : QREMOTE
- Name : OS2.REMOTEQ
- Name on remote system : OS2.LOCALQ
- Remote system name : OS2
- Transmission queue : OS2
- Disposition : GROUP

Sender Channel
- Channel name : MVS.OS2.SNA
- Transport type : L (LU6.2)
- Transmission queue name : OS2
- Connection name : M2
- Disposition : GROUP

WebSphere MQ for z/OS shared receiver-channel definitions using LU 6.2

Local Queue
- Object type : QLOCAL
- Name : QSG.SHAREDQ
- Usage : N (Normal)
- Disposition : SHARED

Receiver Channel
- Channel name : OS2.QSG.SNA
- Disposition : GROUP
WebSphere MQ for z/OS shared sender-channel definitions using TCP

Local Queue
Object type : QLOCAL
Name : OS2
Usage : X (XmitQ)
Disposition : SHARED

Remote Queue
Object type : QREMOTE
Name : OS2.REMOTEQ
Name on remote system : OS2.LOCALQ
Remote system name : OS2
Transmission queue : OS2
Disposition : GROUP

Sender Channel
Channel name : QSG.OS2.TCP
Transport type : T (TCP)
Transmission queue name : OS2
Connection name : os2.tcpip.hostname
Disposition : GROUP

WebSphere MQ for z/OS shared receiver-channel definitions using TCP

Local Queue
Object type : QLOCAL
Name : QSG.SHAREDQ
Usage : N (Normal)
Disposition : SHARED

Receiver Channel
Channel name : OS2.QSG.TCP
Disposition : GROUP
This example illustrates the preparations needed to allow an application using queue manager QM3 to put a message on a queue in a queue-sharing group that has queue members QM4 and QM5.

It is recommended that you are familiar with the example in Chapter 27, “Message channel planning example for z/OS” on page 377 before trying this example.

What this example shows

This example shows the WebSphere MQ commands (MQSC) that you can use in WebSphere MQ for z/OS for distributed queuing with queue-sharing groups. This example expands the payroll query scenario of the example in Chapter 27, “Message channel planning example for z/OS” on page 377 to show how to add higher availability of query processing by adding more serving applications to serve a shared queue.

The payroll query application is now connected to queue manager QM3 and puts a query to the remote queue ‘PAYROLL QUERY’ defined on QM3. This remote queue definition resolves to the shared queue ‘PAYROLL’ hosted by the queue managers in the queue-sharing group QSG1. The payroll processing application now has two instances running, one connected to QM4 and one connected to QM5.
All three queue managers are assumed to be running on z/OS. In the example definitions for TCP/IP, QM4 has a host name of MVSIP01 and QM5 has a host name of MVSIP02. Both queue managers are listening on port 1414 and have registered to use WLM/DNS. The generic address that WLM/DNS provides for this group is QSG1.MVSIP. QM3 has a host address of 9.20.9.31 and is listening on port 1411.

In the example definitions for LU6.2, QM3 is listening on a symbolic luname called LUNAME1. The name of the generic resource defined for VTAM for the lunames listened on by QM4 and QM5 is LUQSG1. The example assumes that these are already defined on your z/OS system and are available for use. To define them see “Defining yourself to the network using generic resources” on page 450.
Queue-sharing group definitions

Producing the following object definitions for one member of the queue-sharing group makes them available to all the other members.

Queue managers QM4 and QM5 are members of the queue sharing group. The definitions produced for QM4 are also available for QM5.

It is assumed that the coupling facility list structure is called ‘APPLICATION1’. If it is not called ‘APPLICATION1’, you must use your own coupling facility list structure name for the example.

Shared objects
The shared object definitions are stored in DB2 and their associated messages are stored within the Coupling Facility.

```
DEFINE QLOCAL(PAYROLL) QSGDISP(SHARED) REPLACE PUT(ENABLED) GET(ENABLED) +
CFSTRUCT(APPLICATION1) +
DESCR('Shared queue for payroll details')

DEFINE QLOCAL(QM3) QSGDISP(SHARED) REPLACE USAGE(XMITQ) PUT(ENABLED) +
CFSTRUCT(APPLICATION1) +
DESCR('Transmission queue to QM3') TRIGGER TRIGTYPE(FIRST) +
TRIGDATA(QSG1.TO.QM3) GET(ENABLED) INITQ(SYSM.CHANNEL.INITQ)
```

Group objects
The group object definitions are stored in DB2, and each queue manager in the queue-sharing group creates a local copy of the defined object.

Sender channel definition:

For a TCP/IP connection:

```
DEFINE CHANNEL(QSG1.TO.QM3) CHLTYPE(SDR) QSGDISP(GROUP) TRPTYPE(TCP) +
REPLACE DESCR('Sender channel to QM3') XMITQ(QM3) +
CONNAME('9.20.9.31(1411)')
```

For an LU 6.2 connection:

```
DEFINE CHANNEL(QSG1.TO.QM3) CHLTYPE(SDR) QSGDISP(GROUP) TRPTYPE(LU62) +
REPLACE DESCR('Sender channel to QM3') XMITQ(QM3) +
CONNAME('LU0001')
```

Receiver channel definition:

For a TCP/IP connection:

```
DEFINE CHANNEL(QM3.TO.QSG1) CHLTYPE(RCVR) TRPTYPE(TCP) +
REPLACE DESCR('Receiver channel from QM3') QSGDISP(GROUP)
```

For an LU 6.2 connection:

```
DEFINE CHANNEL(QM3.TO.QSG1) CHLTYPE(RCVR) TRPTYPE(LU62) +
REPLACE DESCR('Receiver channel from QM3') QSGDISP(GROUP)
```

Queue manager QM3 example

QM3 is not a member of the queue-sharing group.

The following object definitions allow it to put messages to a queue in the queue-sharing group.
Sender channel definition
The conname for this channel is the generic address of the queue-sharing group.

For a TCP/IP connection:
DEFINE CHANNEL(QM3.TO.QSG1) CHLTYPE(SDR) TRPTYPE(TCP) +
REPLACE DESCR('Sender channel to QSG1') XMITQ(QSG1) +
CONNAME('QSG1.MVSIP(1414)')

For an LU 6.2 connection:
DEFINE CHANNEL(QM3.TO.QSG1) CHLTYPE(SDR) TRPTYPE(LU62) +
REPLACE DESCR('Sender channel to QSG1') XMITQ(QSG1) +
CONNAME('LUQSG1) TPNAME('MQSERIES') MODENAME('#INTER')

Remaining definitions
These definitions are required for the same purposes as those in the first example.
DEFINE QREMOTE(PAYROLL.QUERY) DESCR('Remote queue for QSG1') REPLACE +
PUT(ENABLED) XMITQ(QSG1) RNAME(APPL) QRMNAME(QSG1)

DEFINE QLOCAL(QSG1) DESCR('Transmission queue to QSG1') REPLACE +
USAGE(XMITQ) PUT(ENABLED) GET(ENABLED) TRIGGER TRIGTYPE(FIRST) +
TRIGDATA(QM3.TO.QSG1) INITQ(SYSTEM.CHANNEL.INITQ)

DEFINE CHANNEL(QSG1.TO.QM3) CHLTYPE(RCVR) TRPTYPE(TCP) +
REPLACE DESCR('Receiver channel from QSG1')

DEFINE CHANNEL(QSG1.TO.QM3) CHLTYPE(RCVR) TRPTYPE(LU62) +
REPLACE DESCR('Receiver channel from QSG1')

DEFINE QLOCAL(PAYROLL.REPLY) REPLACE PUT(ENABLED) GET(ENABLED) +
DESCR('Reply queue for replies to query messages sent to QSG1')

Running the example
When you have created the required objects, you must do the following.
• Start the channel initiator for all three queue managers.
• Start the listeners for both queue managers in the queue-sharing group.

For a TCP/IP connection each member of the group must have a group listener started that is listening on port 1414.
STA LSTR PORT(1414) IPADDR(MVSIP01) INDISP(GROUP)

The above entry starts the listener on QM4, for example.

For an LU6.2 connection each member of the group must have a group listener started that is listening on a symbolic luname that corresponds to the generic resource LUQSG1.
• Start the listener on QM3
STA LSTR PORT(1411)
Chapter 37. Intra-group queuing

This chapter describes intra-group queuing on WebSphere MQ for z/OS. This function is only available to queue managers defined to a queue-sharing group.

For information about queue-sharing groups, see Chapter 33, “Preparing WebSphere MQ for z/OS for DQM with queue-sharing groups” on page 439.

Concepts

This section describes the concepts of intra-group queuing.

Intra-group queuing (IGQ) can effect potentially fast and less-expensive small message transfer between queue managers within a queue-sharing group (QSG), without the need to define channels between the queue managers. That is, intra-group queuing can be used to deliver, more efficiently, small messages to queues residing on remote queue managers within a queue-sharing group.

Intra-group queuing and the intra-group queuing agent

An IGQ agent is started during queue manager initialization. When applications open and put messages to remote queues, the local queue manager determines whether intra-group queuing should be used for message transfer. If intra-group queuing is to be used, the local queue manager places the message on to the SYSTEM.QSG.TRANSMIT.QUEUE. The IGQ agent on the target remote queue manager retrieves the message and places it on to the destination queue.

The following diagram shows a typical example of intra-group queuing.
The diagram shows:

- IGQ agents running on three queue managers (QMG1, QMG2, and QMG3) that are defined to a queue-sharing group called SQ26.
- Shared transmission queue SYSTEM.QSG.TRANSMIT.QUEUE that is defined in the Coupling Facility (CF).
- A remote queue definition that is defined in queue manager QMG1.
- A local queue that is defined in queue manager QMG2.
- A requesting application (this could be a Message Channel Agent (MCA)) that is connected to queue manager QMG1.
- A server application that is connected to queue manager QMG2.
- A request message being placed on to the SYSTEM.QSG.TRANSMIT.QUEUE.

Terminology

This section describes the terminology of intra-group queuing.

Intra-group queuing

Intra-group queuing can effect potentially fast and less expensive message transfer between queue managers in a queue-sharing group, without the need to define channels.
Shared transmission queue for use by intra-group queuing

Each queue-sharing group has a shared transmission queue called SYSTEM.QSG.TRANSMIT.QUEUE for use by intra-group queuing. If intra-group queuing is enabled, SYSTEM.QSG.TRANSMIT.QUEUE appears in the name resolution path when opening remote queues. When applications (including Message Channel Agents (MCAs)) put messages to a remote queue, the local queue manager determines the eligibility of messages for fast transfer and places them on SYSTEM.QSG.TRANSMIT.QUEUE.

Intra-group queuing agent

The IGQ agent is the task, started at queue manager initialization, that waits for suitable messages to arrive on the SYSTEM.QSG.TRANSMIT.QUEUE. The IGQ agent retrieves suitable messages from this queue and delivers them to the destination queue(s).

The IGQ agent for each queue manager is always started because intra-group queuing is used by the queue manager itself for its own internal processing.

Benefits

This section describes the benefits of intra-group queuing.

**Reduced system definitions**

Intra-group queuing removes the need to define channels between queue managers in a queue-sharing group.

**Reduced system administration**

Because there are no channels defined between queue managers in a queue-sharing group, there is no requirement for channel administration.

**Improved performance**

Because there is only one IGQ agent needed for the delivery of a message to a target queue (instead of two intermediate sender and receiver agents), the delivery of messages using intra-group queuing can be less expensive than the delivery of messages using channels. In intra-group queuing there is only a receiving component, because the need for the sending component has been removed. This saving is because the message is available to the IGQ agent at the destination queue manager for delivery to the destination queue once the put operation at the local queue manager has completed and, in the case of messages put in syncpoint scope, committed.

**Supports migration**

Applications external to a queue-sharing group can deliver messages to a queue residing on any queue manager in the queue-sharing group, while being connected only to a particular queue manager in the queue-sharing group. This is because messages arriving on a receiver channel, destined for a queue on a remote queue manager, can be transparently sent to the destination queue using intra-group queuing. This facility allows applications to be deployed among the queue-sharing group without the need to change any systems that are external to the queue-sharing group.

A typical configuration is illustrated by the following diagram, in which:
• A requesting application connected to queue manager QMG1 needs to send a message to a local queue on queue manager QMG3.
• Queue manager QMG1 is connected only to queue manager QMG2.
• Queue managers QMG2 and QMG3, which were previously connected using channels, are now members of queue-sharing group SQ26.

The flow of operations is as follows:
1. The requesting application puts a message, destined for local queue LQ1 at remote queue manager QMG3, on to remote queue definition RQ1.
2. Queue manager QMG1, running on a Windows NT workstation, places the message on to the transmission queue XQ1.
3. Sender MCA (S) on QM1 transmits the message, via TCP/IP, to the receiver MCA (R) on channel initiator CHINIT2.
4. Receiver MCA (R) on channel initiator CHINIT2 places the message on to the shared transmission queue SYSTEM.QSG.TRANSMIT.QUEUE.
5. IGQ agent on queue manager QMG3 retrieves the message from the SYSTEM.QSG.TRANSMIT.QUEUE and places it on to the target local queue LQ1.
6. The server application retrieves the message from the target local queue and processes it.

Transparent delivery of messages when multi-hopping between queue managers in a queue-sharing group

The above diagram also illustrates the transparent delivery of messages when multi-hopping between queue managers in a queue-sharing group. Messages arriving on a given queue manager within the queue-sharing group, but destined for a queue on another queue manager in the queue-sharing group, can be easily transmitted to the destination queue on the destination queue manager, using intra-group queuing.

Limitations

This section describes the limitations of intra-group queuing.

Messages eligible for transfer using intra-group queuing

Because intra-group queuing makes use of a shared transmission queue that is defined in the Coupling Facility (CF), intra-group queuing is limited to delivering messages of the maximum supported message length for shared queues minus the length of a transmission queue header (MQXQH).
Number of intra-group queuing agents per queue manager

Only one IGQ agent is started per queue manager in a queue-sharing group.

Starting and stopping the intra-group queuing agent

The IGQ agent is started during queue manager initialization and terminated during queue manager shut-down. It is designed to be a long running, self recovering (in the event of abnormal termination), task. If there is an error with the definition of the SYSTEM.QSG.TRANSMIT.QUEUE (for example, if this queue is Get inhibited) the IGQ agent keeps retrying. If the IGQ agent should encounter an error that results in normal termination of the agent while the queue manager is still active, it can be restarted by issuing an ALTER QMGR IGQ(ENABLED) command. This avoids the need to recycle the queue manager.

Getting started

This section describes getting started with intra-group queuing.

Enabling intra-group queuing

To enable intra-group queuing on your queue managers, you need to do the following:

- Define a shared transmission queue called SYSTEM.QSG.TRANSMIT.QUEUE. The definition of this queue can be found in thlqual.SCSQPROCS(CSQ4INSS), the CSQINP2 sample for SYSTEM objects for queue-sharing groups. This queue must be defined with the correct attributes, as stated in thlqual.SCSQPROCS(CSQ4INSS), for intra-group queuing to work properly.
- Because the IGQ agent is always started at queue manager initialization, intra-group queuing is always available for inbound message processing, that is, the IGQ agent processes any messages that are placed on the SYSTEM.QSG.TRANSMIT.QUEUE. However, to enable intra-group queuing for outbound processing, the queue manager attribute IGQ must be set to ENABLED.

Disabling intra-group queuing

To disable intra-group queuing for outbound message transfer, set the queue manager attribute IGQ to DISABLED. Note that if intra-group queuing is disabled for a given queue manager, the IGQ agent on that queue manager can still process inbound messages that have been placed on the SYSTEM.QSG.TRANSMIT.QUEUE by a queue manager that does have intra-group queuing enabled for outbound transfer.

Using intra-group queuing

Once intra-group queuing is enabled, it is available for use and a queue manager uses it whenever possible. That is, when an application puts a message to a remote queue definition, to a fully qualified remote queue, or to a cluster queue, the queue manager determines if the message is eligible to be delivered using intra-group queuing and if it is, places the message on to SYSTEM.QSG.TRANSMIT.QUEUE. There is no need to make changes to user applications, or to application queues, because for eligible messages the queue manager makes use of the SYSTEM.QSG.TRANSMIT.QUEUE, in preference to any other transmission queue.
Configurations

In addition to the typical intra-group queuing configuration described in Figure 91 on page 462, other configurations are possible.

Distributed queuing with intra-group queuing (multiple delivery paths)

For applications that process short messages it might be feasible to configure only intra-group queuing for delivering messages between queue managers in a queue-sharing group. However, for applications that process large (greater than the maximum supported message length for shared queues minus the length of the MQXQH) messages, it may be necessary to configure distributed queuing with intra-group queuing. The following diagram illustrates this configuration.

Open/Put processing

1. It is important to note that when the requesting application opens remote queue RQ1, name resolution occurs for both the non-shared transmission queue XQ1 and the shared transmission queue SYSTEM.QSG.TRANSMIT.QUEUE.

Figure 93. An example configuration
2. When the requesting application puts a message on to the remote queue, the queue manager decides, based on whether intra-group queuing is enabled for outbound transfer and on the message characteristics, whether to put the message to transmission queue XQ1, or to transmission queue SYSTEM.QSG.TRANSMIT.QUEUE. The queue manager places all large messages on to transmission queue XQ1, and all small messages on to transmission queue SYSTEM.QSG.TRANSMIT.QUEUE.

3. If transmission queue XQ1 is full, or is not available, put requests for large messages fail synchronously with a suitable return and reason code. However, put requests for small messages continue to succeed and are placed on transmission queue SYSTEM.QSG.TRANSMIT.QUEUE.

4. If transmission queue SYSTEM.QSG.TRANSMIT.QUEUE is full, or cannot be put to, put requests for small messages fail synchronously with a suitable return and reason code. However, put requests for large messages continue to succeed and are placed on transmission queue XQ1. In this case, no attempt is made to put the small messages on to a transmission queue.

**Flow for large messages**

1. The requesting application puts large messages to remote queue RQ1.
2. Queue manager QMG1 puts the messages on to transmission queue XQ1.
3. Sender MCA (S) on queue manager QMG1 retrieves the messages from transmission queue XQ1 and sends them to queue manager QMG2.
4. Receiver MCA (R) on queue manager QMG2 receives the messages and places them on to destination queue LQ1.
5. The serving application retrieves and subsequently processes the messages from queue LQ1.

**Flow for small messages**

1. The requesting application puts small messages on to remote queue RQ1.
2. Queue manager QMG1 puts the messages on to transmission queue SYSTEM.QSG.TRANSMIT.QUEUE.
3. IGQ on queue manager QMG2 retrieves the messages and places them on to the destination queue LQ1.
4. The serving application retrieves the messages from queue LQ1.

**Points to note about such a configuration**

1. The requesting application does not need to be aware of the underlying mechanism used for the delivery of messages.
2. A potentially faster message delivery mechanism can be achieved for small messages.
3. Multiple paths are available for message delivery (that is, the normal channel route and the intra-group queuing route).
4. The intra-group queuing route, being potentially faster, is selected in preference to the normal channel route. Depending on the message characteristics, message delivery may be divided across the two paths. Hence, messages may be delivered out of sequence (though this is also possible if messages are delivered using only the normal channel route).
5. Once a route has been selected, and messages have been placed on to the respective transmission queues, only the selected route will be used for message delivery. Any unprocessed messages on the SYSTEM.QSG.TRANSMIT.QUEUE are not diverted to transmission queue XQ1.
Clustering with intra-group queuing (multiple delivery paths)

It is possible to configure queue managers so that they are in a cluster as well as in a queue-sharing group. When messages are sent to a cluster queue and the local and remote destination queue managers are in the same queue-sharing group, intra-group queuing is used for the delivery of small messages (using the SYSTEM.QSG.TRANSMIT.QUEUE), while the SYSTEM.CLUSTER.TRANSMIT.QUEUE is used for the delivery of large messages. Also, the SYSTEM.CLUSTER.TRANSMIT.QUEUE is used for the delivery of messages to any queue manager that is in the cluster, but outside the queue-sharing group. The following diagram illustrates this configuration (the channel initiators are not shown).

The diagram shows:
- Four z/OS queue managers QMG1, QMG2, QMG3 and QMG4 configured in a cluster CLUS1.
- Queue managers QMG1, QMG2 and QMG3 configured in a queue-sharing group SQ26.
- IGQ agents running on queue managers QMG2 and QMG3.
- The local SYSTEM.CLUSTER.TRANSMIT.QUEUE defined in QMG1.

Figure 94. An example of clustering with intra-group queuing
• The shared SYSTEM.QSG.TRANSMIT.QUEUE defined in the CF.
• Cluster channels TO.QMG2 (connecting QMG1 to QMG2), TO.QMG3 (connecting QMG1 to QMG3), and TO.QMG4 (connecting QMG1 to QMG4).
• Cluster queue CLUSQ1 being hosted on queue managers QMG2, QMG3 and QMG4.

Assume that the requesting application opens the cluster queue with the MQOO_BIND_NOT_FIXED option, so that the target queue manager for the cluster queue is selected at put time.

If the selected target queue manager is QMG2:
• All large messages put by the requesting application are
  – Put to the SYSTEM.CLUSTER.TRANSMIT.QUEUE on QMG1
  – Transferred to cluster queue CLUSQ1 on QMG2 using cluster channel TO.QMG2
• All small messages put by the requesting application are
  – Put to the shared transmission queue SYSTEM.QSG.TRANSMIT.QUEUE
  – Retrieved by the IGQ agent on QMG2
  – Put to the cluster queue CLUSQ1 on QMG2

If the selected target queue manager is QMG4:
• Because QMG4 is not a member of queue-sharing group SQ26, all messages put by the requesting application are
  – Put to the SYSTEM.CLUSTER.TRANSMIT.QUEUE on QMG1
  – Transferred to cluster queue CLUSQ1 on QMG4 using cluster channel TO.QMG4

**Points to note about such a configuration**
• The requesting application does not need to be aware of the underlying mechanism used for the delivery of messages.
• A potentially faster delivery mechanism is achieved for the transfer of small non-persistent messages between queue managers in a queue-sharing group (even if the same queue managers are in a cluster).
• Multiple paths are available for message delivery (that is, both the cluster route and the intra-group queuing route).
• The intra-group queuing route, being potentially faster, is selected in preference to the cluster route. Depending on the message characteristics, message delivery may be divided across the two paths. Hence, messages may be delivered out of sequence. It is important to note that this will be true irrespective of the MQOO_BIND_\* option specified by the application. Intra-group queuing distributes messages in the same way as clustering does, depending on whether the MQOO_BIND_NOT_FIXED, MQOO_BIND_ON_OPEN, or MQOO_BIND_AS_Q_DEF is specified on open.
• Once a route has been selected, and messages have been placed on to the respective transmission queues, only the selected route is used for message delivery. Any unprocessed messages on the SYSTEM.QSG.TRANSMIT.QUEUE are not diverted to the SYSTEM.CLUSTER.TRANSMIT.QUEUE.

**Clustering, intra-group queuing and distributed queuing**
It is possible to configure a queue manager that is a member of a cluster as well as a queue-sharing group and is connected to a distributed queue manager using a
sender/receiver channel pair. This configuration is a combination of distributed queuing with intra-group queuing, described in "Distributed queuing with intra-group queuing (multiple delivery paths)” on page 466, and clustering with intra-group queuing, described in "Clustering with intra-group queuing (multiple delivery paths)” on page 468.

Messages

This section describes the messages put to the SYSTEM.QSG.TRANSMIT.QUEUE.

Message structure

Like all other messages that are put to transmission queues, messages that are put to the SYSTEM.QSG.TRANSMIT.QUEUE are prefixed with the transmission queue header (MQXQH).

Message persistence

In WebSphere MQ Version 5 Release 3, shared queues support both persistent and non-persistent messages.

If the queue manager terminates while the IGQ agent is processing non-persistent messages, or if the IGQ agent terminates abnormally while in the middle of processing messages, non-persistent messages being processed may be lost. Applications must make arrangements for the recovery of non-persistent messages if their recovery is required.

If a put request for a non-persistent message, issued by the IGQ agent, fails unexpectedly, the message being processed is lost.

Delivery of messages

The IGQ agent retrieves and delivers all non-persistent messages outside of syncpoint scope, and all persistent messages within syncpoint scope. In this case, the IGQ agent acts as the syncpoint coordinator.

Batching of messages

The IGQ agent uses a fixed batchsize of 50 messages. Any persistent messages retrieved within a batch will be committed at intervals of 50 messages. The agent commits a batch consisting of persistent messages when there are no more messages available for retrieval on the SYSTEM.QSG.TRANSMIT.QUEUE.

Message size

The maximum size of message that can be put to the SYSTEM.QSG.TRANSMIT.QUEUE is the maximum supported message length for shared queues minus the length of a transmission queue header (MQXQH).

Default message persistence and default message priority

If the SYSTEM.QSG.TRANSMIT.QUEUE is in the queue name resolution path established at open time, then for messages that are put with default persistence and default priority (or with default persistence or default priority), the usual rules are applied in the selection of the queue whose default priority and persistence values are used. (Refer to WebSphere MQ Application Programming Reference for information on the rules of queue selection).
**Undelivered/unprocessed messages**

If an IGQ agent cannot deliver a message to the destination queue, the IGQ agent:
- Honours the MQRO_DISCARD_MSG report option (if the Report options field of the MQMD for the undelivered message indicates that it should) and discards the undelivered message.
- Attempts to place the undelivered message on to the dead letter queue for the destination queue manager, if the message has not already been discarded. The IGQ agent prefixes the message with a dead letter queue header (MQDLH).

If a dead letter queue is not defined, or if an undelivered message cannot be put to the dead letter queue, and if the undelivered message is:
- persistent, the IGQ agent backs out the current batch of persistent messages that it is processing, and enters a state of retry. For more information, see "Retry capability of the intra-group queuing agent" on page 473.
- non-persistent, the IGQ agent discards the message and continues to process the next message.

If a queue manager in a queue-sharing group is terminated before its associated IGQ agent has had time to process all its messages, the unprocessed messages remain on the SYSTEM.QSG.TRANSMIT.QUEUE until the queue manager is next started. The IGQ agent then retrieves and delivers the messages to the destination queues.

If the coupling facility fails before all the messages on the SYSTEM.QSG.TRANSMIT.QUEUE have been processed, any unprocessed non-persistent messages will be lost.

IBM recommends that applications do not put messages directly to transmission queues. If an application does put messages directly to the SYSTEM.QSG.TRANSMIT.QUEUE, the IGQ agent may not be able to process these messages and they will simply remain on the SYSTEM.QSG.TRANSMIT.QUEUE. Users will then have to use their own methods to deal with these unprocessed messages.

**Report messages**

This section describes report messages.

**Confirmation of arrival (COA)/confirmation of delivery (COD) report messages**

COA and COD messages are generated by the queue manager, when intra-group queuing is used.

**Expiry report messages**

Expiry report messages are generated by the queue manager, when intra-group queuing is used.

**Exception report messages**

Depending on the MQRO_EXCEPTION_* report option specified in the Report options field of the message descriptor for the undelivered message, the IGQ agent generates the required exception report and places it on the specified reply-to queue. (Note that intra-group queuing can be used to deliver the exception report to the destination reply-to queue).

The persistence of the report message is the same as the persistence of the undelivered message. If the IGQ agent fails to resolve the name of the destination queue manager, the report message is discarded.
reply-to queue, or if it fails to put the reply message to a transmission queue (for subsequent transfer to the destination reply-to queue) it attempts to put the exception report to the dead letter queue of the queue manager on which the report message is generated. If this is not possible, then if the undelivered message is:

- persistent, the IGQ agent discards the exception report, backs out the current batch of messages and enters a state of retry. For more information, see “Retry capability of the intra-group queuing agent” on page 473.
- non-persistent, the IGQ agent discards the exception report and continues processing the next message on the SYSTEM.QSG.TRANSMIT.QUEUE.

Security

This section describes the security arrangements for intra-group queuing.

Queue manager attributes IGQAUT (IGQ authority) and IGQUSER (IGQ agent user ID) can be set to control the level of security checking that is performed when the IGQ agent opens destination queues.

**Intra-group queuing authority (IGQAUT)**

The IGQAUT attribute can be set to indicate the type of security checks to be performed, and hence to determine the userids to be used by the IGQ agent when it establishes the authority to put messages on to the destination queue.

The IGQAUT attribute is analogous to the PUTAUT attribute that is available on channel definitions.

**Intra-group queuing user identifier (IGQUSER)**

The IGQUSER attribute can be used to nominate a user ID to be used by the IGQ agent when it establishes the authority to put messages on to a destination queue.

The IGQUSER attribute is analogous to the MCAUSER attribute that is available on channel definitions.

Refer to [WebSphere MQ Script (MQSC) Command Reference](#) and [WebSphere MQ Application Programming Reference](#) for more information about the IGQAUT and IGQUSER queue manager attributes. Refer to the chapter on security in [WebSphere MQ for z/OS System Setup Guide](#) for a table of the userids checked for intra-group queuing.

Specific properties

This section describes the specific properties of intra-group queuing.

**Queue name resolution**

Refer to [WebSphere MQ Script (MQSC) Command Reference](#), [WebSphere MQ Application Programming Reference](#) and [WebSphere MQ for z/OS System Setup Guide](#) for details of queue name resolution when intra-group queuing is used.

**Invalidation of object handles (MQRC_OBJECT_CHANGED)**

If the attributes of an object are found to have changed after the object is opened, the queue manager invalidates the object handle with MQRC_OBJECT_CHANGED on its next use.
Intra-group queuing introduces the following new rules for object handle invalidation:

- If the SYSTEM.QSG.TRANSMIT.QUEUE was included in the name resolution path during open processing because intra-group queuing was ENABLED at open time, but intra-group queuing is found to be DISABLED at put time, then the queue manager invalidates the object handle and fails the put request with MQRC_OBJECT_CHANGED.
- If the SYSTEM.QSG.TRANSMIT.QUEUE was not included in the name resolution path during open processing because intra-group queuing was DISABLED at open time, but intra-group queuing is found to be ENABLED at put time, then the queue manager invalidates the object handle and fails the put request with MQRC_OBJECT_CHANGED.
- If the SYSTEM.QSG.TRANSMIT.QUEUE was included in the name resolution path during open processing because intra-group queuing was enabled at open time, but the SYSTEM.QSG.TRANSMIT.QUEUE definition is found to have changed by put time, then the queue manager invalidates the object handle and fails the put request with MQRC_OBJECT_CHANGED.

**Self recovery of the intra-group queuing agent**

In the event that the IGQ agent terminates abnormally, message CSQM067E is issued and the IGQ agent performs self recovery.

**Retry capability of the intra-group queuing agent**

In the event that the IGQ agent encounters a problem accessing the SYSTEM.QSG.TRANSMIT.QUEUE (because it is not defined, for example, or is defined with incorrect attributes, or is inhibited for Gets, or for some other reason), the IGQ agent goes into the state of retry. The IGQ agent observes short and long retry counts and intervals. The values for these counts and intervals, which cannot be changed, are as follows:

<table>
<thead>
<tr>
<th>Constant</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Short retry count</td>
<td>10</td>
</tr>
<tr>
<td>Short retry interval</td>
<td>60 secs = 1 min</td>
</tr>
<tr>
<td>Long retry count</td>
<td>999,999,999</td>
</tr>
<tr>
<td>Long retry interval</td>
<td>1200 secs = 20 min</td>
</tr>
</tbody>
</table>

**The intra-group queuing agent and Serialization**

If there is a failure of a queue manager in a queue-sharing group while the IGQ agent is dealing with uncommitted messages on a shared queue or queues, the IGQ agent ends, and shared queue peer recovery takes place for the failing queue manager. Because shared queue peer recovery is an asynchronous activity, this leaves the possibility for the failing queue manager, and also the IGQ agent for that queue manager, to restart before shared queue peer recovery is complete. This in turn leaves the possibility for any committed messages to be processed ahead of and out of sequence with the messages still being recovered. To ensure that messages are not processed out of sequence, the IGQ agent serializes access to shared queues by issuing the MQCONNX API call.

An attempt, by the IGQ agent to serialize access to shared queues while peer recovery is still in progress might fail. An error message is issued and the IGQ agent is put into retry state. When queue manager peer recovery is complete, for example at the time of the next retry, the IGQ agent can start.
Chapter 38. Example Configuration — WebSphere MQ for z/OS using intra-group queuing

This section describes how a typical payroll query application, that currently uses distributed queuing to transfer small messages between queue managers, could be migrated to exploit queue sharing groups and shared queues.

Three configurations are described to illustrate the use of distributed queuing, intra-group queuing with shared queues, and shared queues. The associated diagrams show only the flow of data in one direction, that is, from queue manager QMG1 to queue manager QMG3.

Configuration 1

Configuration 1 describes how distributed queuing is currently used to transfer messages between queue managers QMG1 and QMG3.

![Figure 95. Configuration 1: z/OS using intra-group queuing](image)

The flow of operations is as follows:

1. A query is entered using the payroll request application connected to queue manager QMG1.
2. The payroll request application puts the query on to remote queue PAYROLL.QUERY. As queue PAYROLL.QUERY resolves to transmission queue QMG2, the query is put on to transmission queue QMG2.
3. Sender channel (S) on queue manager QMG1 delivers the query to the partner receiver channel (R) on queue manager QMG2.
4. Receiver channel (R) on queue manager QMG2 puts the query on to queue PAYROLL on queue manager QMG3. As queue PAYROLL on QMG3 resolves to transmission queue QMG3, the query is put on to transmission queue QMG3.
5. Sender channel (S) on queue manager QMG2 delivers the query to the partner receiver channel (R) on queue manager QMG3.
6. Receiver channel (R) on queue manager QMG3 puts the query on to local queue PAYROLL.
7. The payroll server application connected to queue manager QMG3 retrieves the query from local queue PAYROLL, processes it, and generates a suitable reply.
Configuration 2

Configuration 2 describes how queue-sharing groups and intra-group queuing can be used, with no effect on the back end payroll server application, to transfer messages between queue managers QMG1 and QMG3. This configuration removes the need for channel definitions between queue managers QMG2 and QMG3 because intra-group queuing is used to transfer messages between these two queue managers.

![Configuration 2 Diagram](image_url)

Figure 96. Configuration 2

The flow of operations is as follows:
1. A query is entered using the payroll request application connected to queue manager QMG1.
2. The payroll request application puts the query on to remote queue PAYROLL.QUERY. As queue PAYROLL.QUERY resolves to transmission queue QMG2, the query is put on to transmission queue QMG2.
3. Sender channel (S) on queue manager QMG1 delivers the query to the partner receiver channel (R) on queue manager QMG2.
4. Receiver channel (R) on queue manager QMG2 puts the query on to queue PAYROLL on queue manager QMG3. As queue PAYROLL on QMG3 resolves to shared transmission queue SYSTEM.QSG.TRANSMIT.QUEUE, the query is put on to shared transmission queue SYSTEM.QSG.TRANSMIT.QUEUE.
5. IGQ agent on queue manager QMG3 retrieves the query from shared transmission queue SYSTEM.QSG.TRANSMIT.QUEUE, and puts it on to local queue PAYROLL on queue manager QMG3.
6. The payroll server application connected to queue manager QMG3 retrieves the query from local queue PAYROLL, processes it, and generates a suitable reply.

Note: The payroll query example transfers small messages only. If you need to transfer both persistent and non-persistent messages, a combination of Configuration 1 and Configuration 2 can be established, so that large messages can be transferred using the distributed queuing route, while small messages can be transferred using the potentially faster intra-group queuing route.

Configuration 3

Configuration 3 describes how queue-sharing groups and shared queues can be used, with no effect on the back end payroll server application, to transfer messages between queue managers QMG1 and QMG3.
The flow of operations is:

1. A query is entered using the payroll request application connected to queue manager QMG1.
2. The payroll request application puts the query on to remote queue PAYROLL.QUERY. As queue PAYROLL.QUERY resolves to transmission queue QMG2, the query is put on to transmission queue QMG2.
3. Sender channel (S) on queue manager QMG1 delivers the query to the partner receiver channel (R) on queue manager QMG2.
4. Receiver channel (R) on queue manager QMG2 puts the query on to shared queue PAYROLL.
5. The payroll server application connected to queue manager QMG3 retrieves the query from shared queue PAYROLL, processes it, and generates a suitable reply.

This configuration is certainly the simplest to configure. However, it should be noted that distributed queuing or intra-group queuing would need to be configured to transfer replies (generated by the payroll server application connected to queue manager QMG3) from queue manager QMG3 to queue manager QMG2, and then on to queue manager QMG1. (See “What this example shows” on page 457 for the configuration used to transfer replies back to the payroll request application.)

**Configuration 1 definitions**

The definitions required for Configuration 1 are as follows (please note that the definitions do not take into account triggering, and that only channel definitions for communication using TCP/IP are provided).

**On QMG1**

Remote queue definition

```
DEFINE QREMOTE(PAYROLL.QUERY) DESCR('Remote queue for QMG3') REPLACE +
    PUT(ENABLED) RNAME(PAYROLL) RQMNAME(QMG3) XMITQ(QMG2)
```

Transmission queue definition

```
DEFINE QLOCAL(QMG2) DESCR('Transmission queue to QMG2') REPLACE +
    PUT(ENABLED) USAGE(XMITQ) GET(ENABLED)
```

Sender channel definition (for TCP/IP)

```
DEFINE CHANNEL(QMG1.TO.QMG2) CHLTYPE(SDR) TRPTYPE(TCP) REPLACE +
    DESCR('Sender channel to QMG2') XMITQ(QMG2) CONNAME('MVSQMG2(1415)')
```
This is the place where you should replace MVSQMG2(1415) with your queue manager connection name and port.

Receiver channel definition (for TCP/IP)
DEFINE CHANNEL(QMG2.TO.QMG1) CHLTYPE(RCVR) TRPTYPE(TCP) + REPLACE DESCR('Receiver channel from QMG2')

Reply-to queue definition
DEFINE QLOCAL(PAYROLL.REPLY) REPLACE PUT(ENABLED) GET(ENABLED) + DESCR('Reply queue for replies to payroll queries sent to QMG3')

On QMG2
Transmission queue definition
DEFINE QLOCAL(QMG1) DESCR('Transmission queue to QMG1') REPLACE + PUT(ENABLED) USAGE(XMITQ) GET(ENABLED)
DEFINE QLOCAL(QMG3) DESCR('Transmission queue to QMG3') REPLACE + PUT(ENABLED) USAGE(XMITQ) GET(ENABLED)

Sender channel definitions (for TCP/IP)
DEFINE CHANNEL(QMG2.TO.QMG1) CHLTYPE(SDR) TRPTYPE(TCP) REPLACE + DESCR('Sender channel to QMG1') XMITQ(QMG1) CONNAME('WINTQMG1(1414)')
This is the place where you should replace WINTQMG1(1414) with your queue manager connection name and port.
DEFINE CHANNEL(QMG2.TO.QMG3) CHLTYPE(SDR) TRPTYPE(TCP) REPLACE + DESCR('Sender channel to QMG3') XMITQ(QMG3) CONNAME('MVSQMG3(1416)')
This is the place where you should replace MVSQMG3(1416) with your queue manager connection name and port.

Receiver channel definition (for TCP/IP)
DEFINE CHANNEL(QMG1.TO.QMG2) CHLTYPE(RCVR) TRPTYPE(TCP) + REPLACE DESCR('Receiver channel from QMG1')
DEFINE CHANNEL(QMG3.TO.QMG2) CHLTYPE(RCVR) TRPTYPE(TCP) + REPLACE DESCR('Receiver channel from QMG3')

On QMG3
Local queue definition
DEFINE QLOCAL(PAYROLL) DESCR('Payroll query request queue') REPLACE + PUT(ENABLED) USAGE(NORMAL) GET(ENABLED) SHARE
DEFINE QLOCAL(QMG2) DESCR('Transmission queue to QMG2') REPLACE + PUT(ENABLED) USAGE(XMITQ) GET(ENABLED)

Sender channel definitions (for TCP/IP)
DEFINE CHANNEL(QMG3.TO.QMG2) CHLTYPE(SDR) TRPTYPE(TCP) REPLACE + DESCR('Sender channel to QMG2') XMITQ(QMG2) CONNAME('MVSQMG2(1415)')
This is the place where you should replace MVSQMG2(1415) with your queue manager connection name and port.

Receiver channel definition (for TCP/IP)
DEFINE CHANNEL(QMG2.TO.QMG3) CHLTYPE(RCVR) TRPTYPE(TCP) + REPLACE DESCR('Receiver channel from QMG2')
Configuration 2 definitions

The definitions required for Configuration 2 are as follows (please note that the definitions do not take into account triggering, and that only channel definitions for communication using TCP/IP are provided). It is assumed that queue managers QMG2 and QMG3 are already configured to be members of the same queue-sharing group.

On QMG1

Remote queue definition
DEFINE QREMOTE(PAYROLL.QUERY) DESCR('Remote queue for QMG3') REPLACE +
PUT(ENABLED) RNAME(PAYROLL) RMNAME(QMG3) XMITQ(QMG2)

Transmission queue definition
DEFINE QLOCAL(QMG2) DESCR('Transmission queue to QMG2') REPLACE +
PUT(ENABLED) USAGE(XMITQ) GET(ENABLED)

Sender channel definition (for TCP/IP)
DEFINE CHANNEL(QMG1.TO.QMG2) CHLTYPE(SDR) TRPTYPE(TCP) REPLACE +
DESCR('Sender channel to QMG2') XMITQ(QMG2) CONNAME('MVSQMG2(1415)')

This is the place where you should replace MVSQMG2(1415) with your queue manager connection name and port.

Receiver channel definition (for TCP/IP)
DEFINE CHANNEL(QMG2.TO.QMG1) CHLTYPE(RCVR) TRPTYPE(TCP) +
REPLACE DESCR('Receiver channel from QMG2')

Reply-to queue definition
DEFINE QLOCAL(PAYROLL.REPLY) REPLACE PUT(ENABLED) GET(ENABLED) +
DESCR('Reply queue for replies to payroll queries sent to QMG3')

On QMG2

Transmission queue definition
DEFINE QLOCAL(QMG1) DESCR('Transmission queue to QMG1') REPLACE +
PUT(ENABLED) USAGE(XMITQ) GET(ENABLED)

DEFINE QLOCAL(SYSTEM.QSG.TRANSMIT.QUEUE) QSGDISP(SHARED) +
DESCR('IGQ Transmission queue') REPLACE PUT(ENABLED) USAGE(XMITQ) +
GET(ENABLED) INDXTYPE(CORRELID) CFSTRUCT('APPLICATION1') +
DEFSPAN(SHARED) DEFPSIST(NO)

This is the place where you should replace APPLICATION1 with your defined CF structure name. Also note that this queue, being a shared queue, need only be defined on one of the queue managers in the queue sharing group.

Sender channel definitions (for TCP/IP)
DEFINE CHANNEL(QMG2.TO.QMG1) CHLTYPE(SDR) TRPTYPE(TCP) REPLACE +
DESCR('Sender channel to QMG1') XMITQ(QMG1) CONNAME('WINTQMG1(1414)')

This is the place where you should replace WINTQMG1(1414) with your queue manager connection name and port.

Receiver channel definition (for TCP/IP)
DEFINE CHANNEL(QMG1.TO.QMG2) CHLTYPE(RCVR) TRPTYPE(TCP) +
REPLACE DESCR('Receiver channel from QMG1')
Queue Manager definition
ALTER QMGR IGQ(ENABLED)

On QMG3
Local queue definition
DEFINE QLOCAL(PAYROLL) DESCR('Payroll query request queue') REPLACE +
PUT(ENABLED) USAGE(NORMAL) GET(ENABLED) SHARE

Queue Manager definition
ALTER QMGR IGQ(ENABLED)

Configuration 3 definitions
The definitions required for Configuration 3 are as follows (please note that the
definitions do not take into account triggering, and that only channel definitions
for communication using TCP/IP are provided). It is assumed that queue
managers QMG2 and QMG3 are already configured to be members of the same
queue-sharing group.

On QMG1
Remote queue definition
DEFINE QREMOTE(PAYROLL.QUERY) DESCR('Remote queue for QMG3') REPLACE +
PUT(ENABLED) RNAME(PAYROLL) RQMNAME(QMG3) XMITQ(QMG2)

Transmission queue definition
DEFINE QLOCAL(QMG2) DESCR('Transmission queue to QMG2') REPLACE +
PUT(ENABLED) USAGE(XMITQ) GET(ENABLED)

Sender channel definition (for TCP/IP)
DEFINE CHANNEL(QMG1.TO.QMG2) CHLTYPE(SDR) TRPTYPE(TCP) +
REPLACE DESCR('Sender channel to QMG2') XMITQ(QMG2) CONNAME('MVSQMG2(1415)')

This is the place where you should replace MVSQMG2(1415) with your queue
manager connection name and port.

Receiver channel definition (for TCP/IP)
DEFINE CHANNEL(QMG2.TO.QMG1) CHLTYPE(RCVR) TRPTYPE(TCP) +
REPLACE DESCR('Receiver channel from QMG2')

Reply-to queue definition
DEFINE QLOCAL(PAYROLL.REPLY) REPLACE PUT(ENABLED) GET(ENABLED) +
DESCR('Reply queue for replies to payroll queries sent to QMG3')

On QMG2
Transmission queue definition
DEFINE QLOCAL(QMG1) DESCR('Transmission queue to QMG1') REPLACE +
PUT(ENABLED) USAGE(XMITQ) GET(ENABLED)

Sender channel definitions (for TCP/IP)
DEFINE CHANNEL(QMG2.TO.QMG1) CHLTYPE(SDR) TRPTYPE(TCP) +
REPLACE DESCR('Sender channel to QMG1') XMITQ(QMG1) CONNAME('WINTQMG1(1414)')

This is the place where you should replace WINTQMG1(1414) with your queue
manager connection name and port.

Receiver channel definition (for TCP/IP)
DEFINE CHANNEL (QMG1 TO QMG2) CHTYPE (RCVR) TRPTYPE (TCP) +
REPLACE DESCR ('Receiver channel from QMG1')

Local queue definition
DEFINE QLOCAL (PAYROLL) QSGDISP (SHARED) DESCR ('Payroll query request queue') +
REPLACE PUT (ENABLED) USAGE (NORMAL) GET (ENABLED) SHARE +
DEFSOPT (SHARED) DEFPSIST (NO) CFSTRUCT (APPLICATION1)

This is the place where you should replace APPLICATION1 with your defined CF structure name. Also note that this queue, being a shared queue, need only be defined on one of the queue managers in the queue sharing group.

On QMG3

No definitions are required on QMG3.

Running the example

For Configuration 1:
1. Start queue managers QMG1, QMG2 and QMG3.
2. Start channel initiators for QMG2 and QMG3.
3. Start the listeners on QMG1, QMG2 and QMG3 to listen on ports 1414, 1415 and 1416, respectively.
4. Start sender channel(s) on QMG1, QMG2 and QMG3.
5. Start the payroll query requesting application connected to QMG1.
6. Start the payroll server application connected to QMG3.
7. Submit a payroll query request to QMG3 and wait for the payroll reply.

For Configuration 2:
1. Start queue managers QMG1, QMG2, and QMG3.
2. Start the channel initiator for QMG2.
3. Start the listeners on QMG1, QMG2 to listen on ports 1414 and 1415, respectively.
4. Start the sender channel on QMG1 and QMG2.
5. Start the payroll query requesting application connected to QMG1.
6. Start the payroll server application connected to QMG3.
7. Submit a payroll query request to QMG3 and wait for the payroll reply.

For Configuration 3:
1. Start queue managers QMG1, QMG2 and QMG3.
2. Start the channel initiator for QMG2.
3. Start the listeners on QMG1, QMG2 to listen on ports 1414 and 1415, respectively.
4. Start sender channels on QMG1 and QMG2.
5. Start the payroll query requesting application connected to QMG1.
6. Start the payroll server application connected to QMG3.
7. Submit a payroll query request to QMG3 and wait for the payroll reply.

Expanding the example

The example can be:
• Expanded to make use of channel triggering as well as application (PAYROLL and PAYROLL.REPLY queue) triggering.
• Configured for communication using LU6.2.
• Expanded to configure more queue managers to the queue sharing group. Then the server application can be cloned to run on other queue manager instances to provide multiple servers for the PAYROLL query queue.
• Expanded to increase the number of instances of the payroll query requesting application to demonstrate the processing of requests from multiple clients.
• Expanded to make use of security (IGQAUT and IGQUSER).

# Part 5. DQM in WebSphere MQ for iSeries

## Chapter 39. Monitoring and controlling channels in WebSphere MQ for iSeries

- The DQM channel control function .......................... 485
- Operator commands ........................................ 486
- Getting started ............................................. 488
- Creating objects ............................................. 488
- Creating a channel .......................................... 488
- Starting a channel .......................................... 490
- Selecting a channel .......................................... 492
- Browsing a channel ......................................... 492
- Renaming a channel ......................................... 494
- Work with channel status .................................. 494
- Work-with-channel choices ................................. 497
- Panel choices ................................................ 497
  - F6=Create ................................................. 497
  - 2=Change ................................................ 498
  - 3=Copy .................................................. 498
  - 4=Delete ................................................ 499
  - 5=Display ............................................... 499
  - 8=Work with Status ..................................... 499
  - 13=Ping .................................................. 499
  - Ping with LU 6.2 ........................................ 499
  - 14=Start ................................................ 499
  - 15=End ................................................... 500
  - Stop immediate ......................................... 501
  - Stop controlled ......................................... 501
  - 16=Reset ................................................ 501
  - 17=Resolve .............................................. 501

## Chapter 40. Preparing WebSphere MQ for iSeries

- Creating a transmission queue ............................... 503
- Triggering channels ........................................ 505
- Channel programs .......................................... 507
- Channel states on OS/400 .................................. 508
- Other things to consider .................................... 509
  - Undelivered-message queue ............................. 509
  - Queues in use .......................................... 509
  - Maximum number of channels ............................ 509
  - Security of WebSphere MQ for iSeries objects ...... 509
  - System extensions and user-exit programs .......... 510

## Chapter 41. Setting up communication for WebSphere MQ for iSeries

- Deciding on a connection .................................... 511
- Defining a TCP connection ................................ 511
  - Receiving on TCP ....................................... 512
    - Using the TCP SO_KEEPALIVE option ............... 512
    - Using the TCP listener backlog option .......... 512
  - Defining an LU 6.2 connection ......................... 513
  - Initiating end (Sending) ................................ 514
  - Initiated end (Receiver) ................................ 517
  - Note on Work Management ............................... 519

## Chapter 42. Example configuration - IBM WebSphere MQ for iSeries

- Configuration parameters for an LU 6.2 connection .... 521
  - Configuration worksheet ................................ 521
  - Explanation of terms .................................... 524
    - How to find network attributes .................... 525
    - How to find the value of Resource name ........... 525
  - Establishing an LU 6.2 connection ..................... 526
    - Local node configuration ............................ 526
    - Creating a line description .......................... 526
    - Adding a routing entry ................................ 527
    - Connection to partner node .......................... 528
    - Creating a controller description .................. 528
    - Creating a device description ........................ 529
    - Creating CPI-C side information ...................... 530
    - Adding a communications entry for APPC .......... 530
    - Adding a configuration list entry ................... 531
    - What next? ............................................ 532
    - Establishing a TCP connection ....................... 532
      - Adding a TCP/IP interface .......................... 532
      - Adding a TCP/IP loopback interface ............... 532
      - Adding a default route ................................ 533
      - What next? ........................................... 534
    - WebSphere MQ for iSeries configuration ............. 534
      - Basic configuration ................................. 534
      - Channel configuration ................................ 535
        - WebSphere MQ for iSeries sender-channel definitions using SNA .......................... 537
        - WebSphere MQ for iSeries receiver-channel definitions using SNA .................... 538
        - WebSphere MQ for iSeries sender-channel definitions using TCP .................... 538
        - WebSphere MQ for iSeries receiver-channel definitions using TCP .................... 538
        - Defining a queue .................................... 538
        - Defining a channel .................................. 539

## Chapter 43. Message channel planning example for iSeries

- What the example shows ..................................... 541
- Queue manager QM1 example ................................ 544
- Queue manager QM2 example ................................ 545
- Running the example ....................................... 545
- Expanding this example ..................................... 545

© Copyright IBM Corp. 1993, 2002
DQM in WebSphere MQ for iSeries
Chapter 39. Monitoring and controlling channels in WebSphere MQ for iSeries

Use the DQM commands and panels to create, monitor, and control the channels to remote queue managers. Each queue manager has a DQM program for controlling interconnections to compatible remote queue managers. See "Operator commands" on page 486 for a list of the commands you need when setting up and controlling message channels.

The DQM channel control function

The channel control function provides the interface and function for administration and control of message channels between WebSphere MQ for iSeries and compatible systems. See Figure 28 on page 58 for a conceptual picture.

The channel control function consists of WebSphere MQ for iSeries panels, commands, programs, a sequence number file, and a file for the channel definitions. The following is a brief description of the components of the channel control function:

- The channel definition file (CDF):
  - Is indexed on channel name
  - Holds channel definitions
- The channel commands are a subset of the WebSphere MQ for iSeries set of commands.
  Use the command GO CMDMQM to display the full set of WebSphere MQ for iSeries commands.
- You use channel definition panels, or commands to:
  - Create, copy, display, change, and delete channel definitions
  - Start and stop channels, ping, reset channel sequence numbers, and resolve in-doubt messages when links cannot be re-established
  - Display status information about channels
- Sequence numbers and logical unit of work (LUW) identifiers are stored in the synchronization file, and are used for channel synchronization purposes.
Operator commands

The following table shows the full list of WebSphere MQ for iSeries commands that you may need when setting up and controlling channels. In general, issuing a command results in the appropriate panel being displayed.

The commands can be grouped as follows:

- **Channel commands**
  - CHGMQMCHL, Change MQM Channel
  - CPYMQMCHL, Copy MQM Channel
  - CRTMQMCHL, Create MQM Channel
  - DLTMQMCHL, Delete MQM Channel
  - DSPMQMCHL, Display MQM Channel
  - ENDMQMCHL, End MQM Channel
  - ENDMQMLSR, End MQM Listener
  - PNNMQMCHL, Ping MQM Channel
  - RSTMQMCHL, Reset MQM Channel
  - RSVPMQMCHL, Resolve MQM Channel
  - STRMQMCHL, Start MQM Channel
  - STRMQMCHLI, Start MQM Channel Initiator
  - STRMQMLSR, Start MQM Listener
  - WRKMQMCHL, Work with MQM Channel
  - WRKMQMCHST, Work with MQM Channel Status

- **Cluster commands**
  - RFRMQMCL, Refresh Cluster
  - RSMMQMCLQM, Resume Cluster Queue Manager
  - RSTMQMCL, Reset Cluster
  - SPDMMQMCLQM, Suspend Cluster Queue Manager
  - WRKMQMCL, Work with Clusters

- **Command Server commands**
  - DSPMQMCSVR, Display MQM Command Server
  - ENDMQMCSVR, End MQM Command Server
  - STRMQMCSVR, Start MQM Command Server

- **Data Type Conversion Command**
  - CVTMQMMDTA, Convert MQM Data Type Command

- **Dead-Letter Queue Handler Command**
  - STRMQMDLQ, Start WebSphere MQ Dead-Letter Queue Handler

- **Media Recovery Commands**
  - RCDMQMIMG, Record MQM Object Image
  - RCRMQMOBJ, Recreate MQM Object

- **WebSphere MQ commands**
  - STRMQMMQSC, Start MQSC Commands

- **Name command**
  - DSTMQMOBJN, Display MQM Object Names

- **Namelist commands**
  - CHGMQMNL, Change MQM Namelist
  - CPYMQMNL, Copy MQM Namelist
  - CRTMQMNL, Create MQM Namelist
  - DLTMQMNL, Delete MQM Namelist
  - DSPMQMNL, Display MQM Namelist
  - WRKMQMNL, Work with MQM Namelists

- **Process commands**
  - CHGMQMPRC, Change MQM Process
Operator commands

CPYMQMPRC, Copy MQM Process
CRTMQMPRC, Create MQM Process
DLTMQMPRC, Delete MQM Process
DSPMQMPRC, Display MQM Process
WRKMQMPRC, Work with MQM Processes

• Queue commands
  CHGQM, Change MQM Queue
  CLRMQM, Clear MQM Queue
  CPYQM, Copy MQM Queue
  CRTQM, Create MQM Queue
  DLTQM, Delete MQM Queue
  DSPQM, Display MQM Queue
  WRKMQMM, Work with MQM Messages
  WRKMQM, Work with MQM Queues

• Queue Manager commands
  CCTQM, Connect Message Queue Manager
  CHGQM, Change Message Queue Manager
  CRTQM, Create Message Queue Manager
  DLTQM, Delete Message Queue Manager
  DSCQM, Disconnect Message Queue Manager
  DSPQM, Display Message Queue Manager
  ENDMQM, End Message Queue Manager
  STRQM, Start Message Queue Manager
  WRKMQM, Work with Message Queue managers

• Security commands
  DSPQMMAUT, Display MQM Object Authority
  GRTQMMAUT, Grant MQM Object Authority
  RVKQMMAUT, Revoke MQM Object Authority

• Trace commands
  TRCMQ, Trace MQM Job

• Transaction commands
  RSVQMTRN, Resolve MQSeries Transaction
  WRKMQTRN, Display MQSeries Transaction

• Trigger Monitor commands
  STRQMTRN, Start Trigger Monitor
Getting started

Use these commands and panels to:
1. Define message channels and associated objects
2. Monitor and control message channels

By using the F4=Prompt key, you can specify the relevant queue manager. If you do not use the prompt, the default queue manager is assumed. With F4=Prompt, an additional panel is displayed where you may enter the relevant queue manager name and sometimes other data.

The objects you need to define with the panels are:
- Transmission queues
- Remote queue definitions
- Queue manager alias definitions
- Reply-to queue alias definitions
- Reply-to local queues
- Processes for triggering (MCAs)
- Message channel definitions

See Chapter 2, “Making your applications communicate” on page 17 for more discussion on the concepts involved in the use of these objects.

Channels must be completely defined, and their associated objects must exist and be available for use, before a channel can be started. This chapter shows you how to do this.

In addition, the particular communication link for each channel must be defined and available before a channel can be run. For a description of how LU 6.2 and TCP/IP links are defined, see the particular communication guide for your installation.

Creating objects

Use the CRTMQMQ command to create the queue and alias objects, such as: transmission queues, remote queue definitions, queue manager alias definitions, reply-to queue alias definitions, and reply-to local queues.

For a list of default objects, see the WebSphere MQ for iSeries V5.3 System Administration book.

Creating a channel

To create a new channel:
1. Use F6 from the Work with MQM Channels panel (the second panel that displays channel details).
   Alternatively, use the CRTMQMCHL command from the command line.
   Either way, this displays the Create Channel panel. Type:
   - The name of the channel in the field provided
   - The channel type for this end of the link
2. Press Enter.

Note: You are strongly recommended to name all the channels in your network uniquely. As shown in Table 1 on page 30, including the source and target queue manager names in the channel name is a good way to do this.
Creating a channel

Your entries are validated and errors are reported immediately. Correct any errors and continue.

You are presented with the appropriate channel settings panel for the type of channel you have chosen. Fill in the fields with the information you have gathered previously. See Appendix A, “Channel planning form” on page 699 for an example of how you might want to gather information. Press Enter to create the channel.

You are provided with help in deciding on the content of the various fields in the descriptions of the channel definition panels in the help panels, and in Chapter 6, “Channel attributes” on page 77.

Figure 98. Create channel (1)

Figure 99. Create channel (2)
Starting a channel

Listeners are valid for TCP only. For SNA listeners, you must configure your communications subsystem.

For applications to be able to exchange messages you must start a listener program for inbound connections using the STRMQMLSR command.

For outbound connections you must start the channel in one of the following ways:

**Figure 100. Create channel (3)**

<table>
<thead>
<tr>
<th>Option</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Message exit user data</td>
<td>+SYSDFTCHL</td>
</tr>
<tr>
<td>Convert message</td>
<td>+SYSDFTCHL+YES, +NO, +SYSDFTCHL</td>
</tr>
<tr>
<td>Sequence number wrap</td>
<td>999999999999, 100-9999999999, +SYSDFTCHL</td>
</tr>
<tr>
<td>Maximum message length</td>
<td>4194304, 0-4194304, +SYSDFTCHL</td>
</tr>
<tr>
<td>Heartbeat interval</td>
<td>300, 0-999999999, +SYSDFTCHL</td>
</tr>
<tr>
<td>Non Persistent Message Speed</td>
<td>+FAST, +FAST, +NORMAL, +SYSDFTCHL</td>
</tr>
<tr>
<td>Task User Profile</td>
<td>+SYSDFTCHL Character value, +BLANK...</td>
</tr>
<tr>
<td>Transaction Program Name</td>
<td>+SYSDFTCHL</td>
</tr>
</tbody>
</table>

**Figure 101. Create channel (4)**

**Starting a channel**

Listeners are valid for TCP only. For SNA listeners, you must configure your communications subsystem.

For applications to be able to exchange messages you must start a listener program for inbound connections using the STRMQMLSR command.

For outbound connections you must start the channel in one of the following ways:
Creating a channel

1. Use the CL command STRMQMCHL, specifying the channel name, to start the channel as a process or a thread, depending on the MCATYPE parameter. (If channels are started as threads, they are threads of a channel initiator.)
   
   STRMQMCHL CHLNAME(QM1.TO.QM2) MQNAME(MYQMGR)

2. Use a channel initiator to trigger the channel. One channel initiator is started automatically when the queue manager is started. This can be eliminated by changing the chinit stanza in the qm.ini file for that queue manager.

3. Use the WRKMQMCHL command to begin the Work with Channels panel and choose option 14 to start a channel.
Selecting a channel

To select a channel, use the WRKMQMCHL command to begin at the Work with Channels panel:
1. Move the cursor to the option field at the left of the required channel name.
2. Type an option number.
3. Press Enter to activate your choice.

If you select more than one channel, the options are activated in sequence.

Browsing a channel

To browse the settings of a channel, use the WRKMQMCHL command to begin at the Display Channel panel:
1. Move the cursor to the left of the required channel name.
2. Type option 5 (Display).
3. Press Enter to activate your choice.

If you select more than one channel, they are presented in sequence.

Alternatively, you can use the DSPMQMCHL command from the command line.

This results in the respective Display Channel panel being displayed with details of the current settings for the channel. The fields are described in Chapter 6, "Channel attributes" on page 77.
### Figure 103. Display a TCP/IP channel (1)

<table>
<thead>
<tr>
<th>Display MQM Channel</th>
</tr>
</thead>
<tbody>
<tr>
<td>Channel name ..........: ST.JST.2T01</td>
</tr>
<tr>
<td>Queue Manager Name ....: QMREL</td>
</tr>
<tr>
<td>Channel type ..........: *SDR</td>
</tr>
<tr>
<td>Transport type .......: *TCP</td>
</tr>
<tr>
<td>Text 'description' .....: John's sender to WINSOAI</td>
</tr>
<tr>
<td>Connection name .......: MUSTANG</td>
</tr>
<tr>
<td>Transmission queue ......: WINSOAI</td>
</tr>
<tr>
<td>Message channel agent ...:</td>
</tr>
<tr>
<td>Library ...............:</td>
</tr>
<tr>
<td>Message channel agent user ID : *NONE</td>
</tr>
<tr>
<td>Batch interval ..........: 0</td>
</tr>
<tr>
<td>Batch size ............: 50</td>
</tr>
<tr>
<td>Disconnect interval .....: 6000</td>
</tr>
</tbody>
</table>

F3=Exit  F12=Cancel  F21=Print

### Figure 104. Display a TCP/IP channel (2)

<table>
<thead>
<tr>
<th>Display MQM Channel</th>
</tr>
</thead>
<tbody>
<tr>
<td>Short retry interval .: 60</td>
</tr>
<tr>
<td>Short retry count ....: 10</td>
</tr>
<tr>
<td>Long retry interval ...: 6000</td>
</tr>
<tr>
<td>Long retry count ......: 10</td>
</tr>
<tr>
<td>Security exit ..........:</td>
</tr>
<tr>
<td>Library ...............:</td>
</tr>
<tr>
<td>Security exit user data:</td>
</tr>
<tr>
<td>Send exit .............:</td>
</tr>
<tr>
<td>Library ...............:</td>
</tr>
<tr>
<td>Send exit user data ...:</td>
</tr>
<tr>
<td>Receive exit ..........:</td>
</tr>
<tr>
<td>Library ...............:</td>
</tr>
<tr>
<td>Receive exit user data:</td>
</tr>
<tr>
<td>Message exit ...........:</td>
</tr>
<tr>
<td>Library ...............:</td>
</tr>
<tr>
<td>Message exit user data ..:</td>
</tr>
</tbody>
</table>

More...

F3=Exit  F12=Cancel  F21=Print
Renaming a channel

To rename a message channel, begin at the Work with Channels panel:
1. End the channel.
2. Use option 3 (Copy) to create a duplicate with the new name.
3. Use option 5 (Display) to check that it has been created correctly.
4. Use option 4 (Delete) to delete the original channel.

If you decide to rename a message channel, ensure that both channel ends are renamed at the same time.

Work with channel status

Use the WRKMQMCHST command to bring up the first of three screens showing the status of your channels. You can view the three status screens in sequence when you select Change-view (F11).

Alternatively, selecting option 8 (Work with Status) from the Work with MQM Channels panel also brings up the first status panel.

Work with channel status applies to all message channels. It does not apply to MQI channels other than server-connection channels on WebSphere MQ for iSeries V5.1 and later.

Note: The Work with Channel Status screens only show channels that are active after messages have been sent through the channel and the sequence number has been incremented.
### MQSeries Work with Channel Status

Type options, press Enter.

5=Display 13=Ping 14=Start 15=End 16=Reset 17=Resolve

<table>
<thead>
<tr>
<th>Opt</th>
<th>Name</th>
<th>Connection</th>
<th>Indoubt</th>
<th>Last Seq</th>
</tr>
</thead>
<tbody>
<tr>
<td>CARTS_CORSAIR_CHAN</td>
<td>GBIBM1YA.WINSDDOA1</td>
<td>NO</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>CHLNIC</td>
<td>9.20.2.213</td>
<td>NO</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>FV.CHANNEL.PETER2</td>
<td>9.20.2.201</td>
<td>NO</td>
<td>28</td>
<td></td>
</tr>
<tr>
<td>JST.1.2</td>
<td>9.20.2.213</td>
<td>NO</td>
<td>6225</td>
<td></td>
</tr>
<tr>
<td>MP_MUST_TO_CORS</td>
<td>9.20.2.201</td>
<td>NO</td>
<td>100</td>
<td></td>
</tr>
<tr>
<td>MUSTANG.TO.CORSAIR</td>
<td>GBIBM1YA.WINSDDOA1</td>
<td>NO</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>MP_CORS_TO_MUST</td>
<td>9.20.2.213</td>
<td>NO</td>
<td>101</td>
<td></td>
</tr>
<tr>
<td>JST.2.3</td>
<td>9.5.7.126</td>
<td>NO</td>
<td>32</td>
<td></td>
</tr>
<tr>
<td>ST.JCW.EXIT.2TO1.CHL</td>
<td>9.20.2.213</td>
<td>NO</td>
<td>216</td>
<td></td>
</tr>
</tbody>
</table>

Parameters or command

F3=Exit  F4=Prompt  F5=Refresh  F6=Create  F9=Retrieve  F11=Change view
F12=Cancel  F21=Print

---

**Figure 106. Channel status (1)**

Change the view with F11.

---

### MQSeries Work with Channel Status

Type options, press Enter.

5=Display 13=Ping 14=Start 15=End 16=Reset 17=Resolve

<table>
<thead>
<tr>
<th>Opt</th>
<th>Transmission Queue</th>
<th>LUWID</th>
</tr>
</thead>
<tbody>
<tr>
<td>FV_MKP_TRANS_QUEUE</td>
<td>75147B61A44000FA</td>
<td></td>
</tr>
<tr>
<td>JST.3</td>
<td>75170185D0000133</td>
<td></td>
</tr>
<tr>
<td>PF.WINSDDOA1</td>
<td>75160A3955C00097</td>
<td></td>
</tr>
<tr>
<td>PF.WINSDDOA1</td>
<td>7516C51291400016</td>
<td></td>
</tr>
</tbody>
</table>

Parameters or command

F3=Exit  F4=Prompt  F5=Refresh  F6=Create  F9=Retrieve  F11=Change view
F12=Cancel  F21=Print

---

**Figure 107. Channel status (2)**
The options available in the Work with Channel Status panel are:

<table>
<thead>
<tr>
<th>Menu option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>5=Display</td>
<td>Displays the channel settings.</td>
</tr>
<tr>
<td>13=Ping</td>
<td>Initiates a Ping action, where appropriate.</td>
</tr>
<tr>
<td>14=Start</td>
<td>Starts the channel.</td>
</tr>
<tr>
<td>15=End</td>
<td>Stops the channel.</td>
</tr>
<tr>
<td>16=Reset</td>
<td>Resets the channel sequence number.</td>
</tr>
<tr>
<td>17=Resolve</td>
<td>Resolves an in-doubt channel situation, manually.</td>
</tr>
<tr>
<td>F11=Change view</td>
<td>Cycles around the three status panels.</td>
</tr>
</tbody>
</table>

Note: When using the WRKMQMCHST command, the channel status shown is SAVED channel status not CURRENT channel status. To see CURRENT channel status, use the WRKMQMCHL command.

Work-with-channel choices

The Work with Channels panel is reached with the command WRKMQMCHL, and it allows you to monitor the status of all channels listed, and to issue commands against selected channels.
The options available in the Work with Channel panel are:

<table>
<thead>
<tr>
<th>Menu option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>F6=Create</td>
<td>Creates a channel.</td>
</tr>
<tr>
<td>2=Change</td>
<td>Changes the attributes of a channel.</td>
</tr>
<tr>
<td>3=Copy</td>
<td>Copies the attributes of a channel to a new channel.</td>
</tr>
<tr>
<td>4=Delete</td>
<td>Deletes a channel.</td>
</tr>
<tr>
<td>5=Display</td>
<td>Displays the current settings for the channel.</td>
</tr>
<tr>
<td>8=Work with status</td>
<td>Displays the channel status panels.</td>
</tr>
<tr>
<td>13=Ping</td>
<td>Runs the Ping facility to test the connection to the adjacent system by exchanging a fixed data message with the remote end.</td>
</tr>
<tr>
<td>14=Start</td>
<td>Starts the selected channel, or resets a disabled receiver channel.</td>
</tr>
<tr>
<td>15=End</td>
<td>Requests the channel to close down.</td>
</tr>
<tr>
<td>16=Reset</td>
<td>Requests the channel to reset the sequence numbers on this end of the link. The numbers must be equal at both ends for the channel to start.</td>
</tr>
<tr>
<td>17=Resolve</td>
<td>Requests the channel to resolve in-doubt messages without establishing connection to the other end.</td>
</tr>
</tbody>
</table>

### Panel choices

The following choices are provided in the Work with MQM channels panel and the Work with Channel Status panel.

**F6=Create**

Use the Create option, or enter the Crtmqmchl command from the command line, to obtain the **Create Channel panel**. There are examples of Create Channel panels, starting at Figure 98 on page 489.

With this panel, you create a new channel definition from a screen of fields filled with default values supplied by WebSphere MQ for iSeries. Type the name of the channel, select the type of channel you are creating, and the communication method to be used.

When you press Enter, the panel is displayed. Type information in all the required fields in this panel, and the three pages making up the complete panel, and then save the definition by pressing Enter.

The channel name must be the same at both ends of the channel, and unique within the network. However, you should restrict the characters used to those that are valid for WebSphere MQ for iSeries object names; see **Chapter 6, “Channel attributes” on page 77**.

All panels have default values supplied by WebSphere MQ for iSeries for some fields. You can customize these values, or you can change them when you are creating or copying channels. To customize the values, see the **WebSphere MQ for iSeries System Administration**.

You can create your own set of channel default values by setting up dummy channels with the required defaults for each channel type, and copying them each time you want to create new channel definitions.

**Table 44 on page 498** shows the channel attributes for each type of channel. See **Chapter 6, “Channel attributes” on page 77** for details about the fields.
Table 44. Channel attribute fields per message channel type

<table>
<thead>
<tr>
<th>Attribute field</th>
<th>Sender</th>
<th>Server</th>
<th>Receiver</th>
<th>Requester</th>
</tr>
</thead>
<tbody>
<tr>
<td>Batch size</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Channel name</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Channel type</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Connection name</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Context</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Disconnect interval</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Heartbeat interval</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Long retry wait interval</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Long retry count</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Maximum message length</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Message channel agent name</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Message exit user data</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Message retry exit count</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Message retry exit data</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Message retry exit interval</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Message retry exit name</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Nonpersistent message speed</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Receive exit</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Receive exit user data</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Security exit</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Security exit user data</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Send exit</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Send exit user data</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sequence number wrap</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Short retry wait interval</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Short retry count</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Transport type</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Transmission queue</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Message exit</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**2=Change**

Use the Change option, or the CHGMQMCHL command, to change an existing channel definition, except for the channel name. Simply type over the fields to be changed in the channel definition panel, and then save the updated definition by pressing Enter.

**3=Copy**

Use the Copy option, or the CPYMQMCHL command, to copy an existing channel. The Copy panel enables you to define the new channel name. However, you should restrict the characters used to those that are valid for WebSphere MQ for iSeries object names; see the *WebSphere MQ for iSeries System Administration*.

Press Enter on the Copy panel to display the details of current settings. You can change any of the new channel settings. Save the new channel definition by pressing Enter.
4=Delete
Use the Delete option to delete the selected channel. A panel is displayed to confirm or cancel your request.

5=Display
Use the Display option to display the current definitions for the channel. This choice displays the panel with the fields showing the current values of the parameters, and protected against user input.

8=Work with Status
The status column tells you whether the channel is active or inactive, and is displayed continuously in the Work with MQM Channels panel. Use option 8 (Work with Status) to see more status information displayed. Alternatively, this can be displayed from the command line with the WRKMQMCHST command. See "Work with channel status" on page 494.

- Channel name
- Communication connection name
- In-doubt status of channel (where appropriate)
- Last sequence number
- Transmission queue name (where appropriate)
- The in-doubt identifier (where appropriate)
- The last committed sequence number
- Logical unit of work identifier

13=Ping
Use the Ping option to exchange a fixed data message with the remote end. This gives some confidence to the system supervisor that the link is available and functioning.

Ping does not involve the use of transmission queues and target queues. It uses channel definitions, the related communication link, and the network setup.

It is available from sender and server channels, only. The corresponding channel is started at the far side of the link, and performs the start up parameter negotiation. Errors are notified normally.

The result of the message exchange is presented in the Ping panel for you, and is the returned message text, together with the time the message was sent, and the time the reply was received.

Ping with LU 6.2
When Ping is invoked in WebSphere MQ for iSeries, it is run with the USERID of the user requesting the function, whereas the normal way that a channel program is run is for the QMQM USERID to be taken for channel programs. The USERID flows to the receiving side and it must be valid on the receiving end for the LU 6.2 conversation to be allocated.

14=Start
The Start option is available for sender, server, and requester channels. It should not be necessary where a channel has been set up with queue manager triggering.

The Start option is also used for receiver channels that have a DISABLED or STOPPED status. Starting a receiver channel that is in DISABLED or STOPPED state resets the channel and allows it to be started from the remote channel.
When started, the sending MCA reads the channel definition file and opens the transmission queue. A channel start-up sequence is executed, which remotely starts the corresponding MCA of the receiver or server channel. When they have been started, the sender and server processes await messages arriving on the transmission queue and transmit them as they arrive.

When you use triggering, you will need to start the continuously running trigger process to monitor the initiation queue. The STRMQMCHLI command can be used for this.

At the far end of a channel, the receiving process may be started in response to a channel startup from the sending end. The method of doing this is different for LU 6.2 and TCP/IP connected channels:

- LU 6.2 connected channels do not require any explicit action at the receiving end of a channel.
- TCP connected channels require a listener process to be running continuously. This process awaits channel startup requests from the remote end of the link and starts the process defined in the channel definitions for that connection. When the remote machine is an iSeries, you can use the STRMQMLSR command for this.

Use of the Start option always causes the channel to re-synchronize, where necessary.

For the start to succeed:

- Channel definitions, local and remote must exist. If there is no appropriate channel definition for a receiver or server-connection channel, a default one is created automatically if the channel is auto-defined. See “Channel auto-definition exit program” on page 583.
- The transmission queue must exist, be enabled for GETs, and have no other channels using it.
- MCAs, local and remote, must exist.
- The communication link must be available.
- The queue managers must be running, local and remote.
- The message channel must be inactive.

To transfer messages, remote queues and remote queue definitions must exist.

A message is returned to the panel confirming that the request to start a channel has been accepted. For confirmation that the Start process has succeeded, check the system log, or press F5 (refresh the screen).

15=End

Use the End option to request the channel to stop activity. The channel will not send any more messages until the operator starts the channel again. (For information about restarting stopped channels, see “Restarting stopped channels” on page 69.)

You can select the type of stop you require if you press F4 before Enter. You can choose IMMEDIATE, or CONTROLLED.
Stop immediate
Normally, this option should not be used. It terminates the channel process. The channel does not complete processing the current batch of messages, and cannot, therefore, leave the channel in doubt. In general, it is recommended that the operators use the controlled stop option.

Stop controlled
This choice requests the channel to close down in an orderly way; the current batch of messages is completed, and the syncpoint procedure is carried out with the other end of the channel.

16=Reset
The Reset option changes the message sequence number. Use it with care, and only after you have used the Resolve option to resolve any in-doubt situations. This option is available only at the sender or server channel. The first message starts the new sequence the next time the channel is started.

17=Resolve
Use the Resolve option when messages are held in-doubt by a sender or server, for example because one end of the link has terminated, and there is no prospect of it recovering. The Resolve option accepts one of two parameters: BACKOUT or COMMIT. Backout restores messages to the transmission queue, while Commit discards them.

The channel program does not try to establish a session with a partner. Instead, it determines the logical unit of work identifier (LUWID) which represents the in-doubt messages. It then issues, as requested, either:
- BACKOUT to restore the messages to the transmission queue; or
- COMMIT to delete the messages from the transmission queue.

For the resolution to succeed:
- The channel must be inactive
- The channel must be in doubt
- The channel type must be sender or server
- The channel definition, local, must exist
- The queue manager must be running, local
Panel choices
Chapter 40. Preparing WebSphere MQ for iSeries

This chapter describes the WebSphere MQ for iSeries preparations required before DQM can be used. Communication preparations are described in Chapter 41, “Setting up communication for WebSphere MQ for iSeries” on page 511.

Before a channel can be started, the transmission queue must be defined as described in this chapter, and must be included in the message channel definition.

In addition, where needed, the triggering arrangement must be prepared with the definition of the necessary processes and queues.

Creating a transmission queue

You define a local queue with the Usage field attribute set to *TMQ, for each sending message channel.

If you want to make use of remote queue definitions, use the same command to create a queue of type *RMT, and Usage of *NORMAL.

To create a transmission queue, use the CRTMQMQ command from the command line to present you with the first queue creation panel; see Figure 109.

```
Create MQM Queue (CRTMQMQ)

Type choices, press Enter.
Queue name . . . . . . . .
Queue type . . . . . . . . . ___ +ALS, +LCL, +MDL, +RMT
Message Queue Manager name . . +DFT___________________________

F3=Exit  F4=Prompt  F5=Refresh  F12=Cancel  F13=How to use this display
F24=More keys
```

Figure 109. Create a queue (1)

Type the name of the queue and specify the type of queue that you wish to create: Local, Remote, or Alias. For a transmission queue, specify Local (+LCL) on this panel and press Enter.
Creating a transmission queue

You are presented with the second page of the Create MQM Queue panel; see Figure 110.

**Create MQM Queue (CRTMQMQ)**

Type choices, press Enter.

Queue name ............... > HURS.2.HURS.PRIORIT
Queue type ............... > *LCL, *ALS, *MDL, *RMT
Message Queue Manager name .... *DFT
Replace ............... *NO, *YES
Text 'description' ....... ' '
Put enabled ............... *YES, *NO, *YES
Default message priority .... 0, *SYSDFTQ
Default message persistence . . *NO, *YES
Process name ............... ' '
Triggering enabled ......... *NO, *YES
Get enabled ............... *YES, *NO, *YES
Sharing enabled ............. *YES, *NO, *YES

Change any of the default values shown. Press page down to scroll to the next screen; see Figure 111.

**Create MQM Queue (CRTMQMQ)**

Type choices, press Enter.

Default share option ...... *YES, *SYSDFTQ, *NO, *YES
Harden backout count ......... *NO, *SYSDFTQ, *NO, *YES
Trigger type ............... *FIRST, *SYSDFTQ, *FIRST, *ALL...
Trigger depth ............... 1, 1-999999999, *SYSDFTQ
Trigger message priority .... 0, 0-9, *SYSDFTQ
Trigger data ............... ' '
Retention interval ........ 999999999 0-999999999, *SYSDFTQ
Maximum queue depth ....... 5000 1-24000, *SYSDFTQ
Maximum message length ..... 4194304 0-4194304, *SYSDFTQ
Backout threshold .......... 0 0-999999999, *SYSDFTQ
Backout requeue queue ....... ' '
Initiation queue ............... ' '

Type *TMQ, for transmission queue, in the Usage field of this panel, and change any of the default values shown in the other fields.

**Figure 110. Create a queue (2)**

**Figure 111. Create a queue (3)**
When you are satisfied that the fields contain the correct data, press Enter to create the queue.

### Triggering channels

An overview of triggering is given in "Triggering channels" on page 20, while it is described in depth in the *WebSphere MQ Application Programming Guide*. This section provides you with information specific to WebSphere MQ for iSeries.

Triggering in WebSphere MQ for iSeries is implemented with the channel initiator process, which is started with the STRMQMCHLI command that specifies the name of the initiation queue. For example:

```
STRMQMCHLI QNAME(MYINITQ)
```

You need to set up the transmission queue for the channel specifying TRIGGER and specifying the channel name in the TRIGDATA field: For example:

```
CRTMQMQ QNAME(MYXMITQ) QTYPE(*LCL) MQMNAME(MYQMGR) +
        PRCNAME(MYPROCESS) TRGENBL(*YES) INITQNAME(MYINITQ) +
        USAGE(*TMQ)
```

Then define an initiation queue.

```
CRTMQMQ QNAME(MYINITQ) MQMNAME(MYQMGR)
```

Next you define a process in WebSphere MQ for iSeries naming the MCA sender program, as the program to be triggered when messages arrive on the transmission queue. Type CRTMQMPRC on the command line to display the Create Process panel. Alternatively, select F6 (Create) from the Work with MQM Process panel. See Figure 113 on page 506 for the first page of the Create Process panel. The *WebSphere MQ for iSeries System Administration* contains details of defining processes to be triggered.
Triggering channels

**Create MQM Process (CRTMQMPRC)**

Type choices, press Enter.

- **Process Name** ....... _______________________________________
- **Message Queue Manager name** . . *DFT_____________________________
- **Replace** ............ *NO, *YES
- **Text 'description'** ....... > 'Triggers hursley.to.hursley.normal'
- **Application type** ........ *OS400
- **Application identifier** . . > 'AMQRMCLA

**User data** .............. > *SYSDFTPRC

More...

F3=Exit  F4=Prompt  F5=Refresh  F12=Cancel  F13=How to use this display
F24=More keys

*Figure 113. Create process (1)*

1. Type the name of the process definition in the field provided.
2. Enter a description in the **Text 'description'** field.
3. Set **Application type** to *OS400.
4. Set **Application identifier** to AMQRMCLA.
5. Set **User data** to the channel name so as to associate this definition with the transmission queue belonging to the channel.
6. Page down to show the second page (see page 507) and insert any environment data.
There are different types of channel programs (MCAs) available for use at the channels. The names are contained in the following table.

**Table 45. Program and transaction names**

<table>
<thead>
<tr>
<th>Program name</th>
<th>Direction of connection</th>
<th>Communication</th>
</tr>
</thead>
<tbody>
<tr>
<td>AMQCRSTA</td>
<td>Inbound</td>
<td>TCP</td>
</tr>
<tr>
<td>AMQCRS6A</td>
<td>Inbound</td>
<td>LU 6.2</td>
</tr>
<tr>
<td>AMQRMCLA</td>
<td>Outbound</td>
<td>Any</td>
</tr>
</tbody>
</table>
Channel states on OS/400

Channel states are displayed on the Work with Channels panel (described in Figure 102 on page 492). There are some differences between the names of channel states on different versions of WebSphere MQ for iSeries. In the following table, the state names shown for V4R2 correspond to the channel states described in Figure 30 on page 63. As shown in the table, some of these states have different names, or do not exist for earlier versions.

Table 46. Channel states on OS/400

<table>
<thead>
<tr>
<th>State name (V3R6)</th>
<th>State name (V3R2, V3R7, V4R2, V5R1)</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>-</td>
<td>STARTING</td>
<td>Channel is ready to begin negotiation with target MCA</td>
</tr>
<tr>
<td>BINDING</td>
<td>BINDING</td>
<td>Establishing a session and initial data exchange</td>
</tr>
<tr>
<td>REQUESTING</td>
<td>REQUESTING</td>
<td>Requester channel initiating a connection</td>
</tr>
<tr>
<td>READY</td>
<td>RUNNING</td>
<td>Transferring or ready to transfer</td>
</tr>
<tr>
<td>PAUSED</td>
<td>PAUSED</td>
<td>Waiting for message-retry interval</td>
</tr>
<tr>
<td>CLOSING</td>
<td>STOPPING</td>
<td>Establishing whether to retry or stop</td>
</tr>
<tr>
<td>RETRYING</td>
<td>RETRYING</td>
<td>Waiting until next retry attempt</td>
</tr>
<tr>
<td>DISABLED</td>
<td>STOPPED</td>
<td>Channel stopped because of an error or because an end-channel command is issued</td>
</tr>
<tr>
<td>STOPPED</td>
<td>INACTIVE</td>
<td>Channel ended processing normally or channel never started</td>
</tr>
<tr>
<td>-</td>
<td>*None</td>
<td>No state (for server-connection channels only)</td>
</tr>
</tbody>
</table>

Note: The state *None applies only to V3R2 and V3R7.
Other things to consider

Here are some other topics that you should consider when preparing WebSphere MQ for distributed queue management.

**Undelivered-message queue**

It is advisable that you have an application available to process the messages arriving on the undelivered-message queue (also known as the dead-letter queue or DLQ). The program could be triggered, or run at regular intervals. For more details, see the *WebSphere MQ for iSeries System Administration* and the *WebSphere MQ Application Programming Guide*.

**Queues in use**

MCAs for receiver channels may keep the destination queues open even when messages are not being transmitted; this results in the queues appearing to be “in use”.

**Maximum number of channels**

You can specify the maximum number of channels allowed in your system and the maximum number that can be active at one time. You do this in the `qm.ini` file in directory `QIBM/UserData/mqm/qmgrs/queue manager name`. See Appendix D, “Configuration file stanzas for distributed queuing” on page 715.

**Security of WebSphere MQ for iSeries objects**

This section deals with remote messaging aspects of security.

You need to provide users with authority to make use of the WebSphere MQ for iSeries facilities, and this is organized according to actions to be taken with respect to objects and definitions. For example:

- Queue managers can be started and stopped by authorized users
- Applications need to connect to the queue manager, and have authority to make use of queues
- Message channels need to be created and controlled by authorized users
Other things to consider

The message channel agent at a remote site needs to check that the message being delivered has derived from a user with authority to do so at this remote site. In addition, as MCAs can be started remotely, it may be necessary to verify that the remote processes trying to start your MCAs are authorized to do so. There are three possible ways for you to deal with this:

1. Decree in the channel definition that messages must contain acceptable context authority, otherwise they will be discarded.

2. Implement user exit security checking to ensure that the corresponding message channel is authorized. The security of the installation hosting the corresponding channel ensures that all users are properly authorized, so that you do not need to check individual messages.

3. Implement user exit message processing to ensure that individual messages are vetted for authorization.

Here are some facts about the way WebSphere MQ for iSeries operates security:

- Users are identified and authenticated by iSeries
- Queue manager services invoked by applications are run with the authority of the queue manager user profile, but in the user’s process
- Queue manager services invoked by user commands are run with the authority of the queue manager user profile

System extensions and user-exit programs

A facility is provided in the channel definition to allow extra programs to be run at defined times during the processing of messages. These programs are not supplied with WebSphere MQ for iSeries, but may be provided by each installation according to local requirements.

In order to run, such programs must have predefined names and be available on call to the channel programs. The names of the exit programs are included in the message channel definitions.

There is a defined control block interface for handing over control to these programs, and for handling the return of control from these programs.

The precise places where these programs are called, and details of control blocks and names, are to be found in Part 7, “Further intercommunication considerations” on page 567.
Chapter 41. Setting up communication for WebSphere MQ for iSeries

DQM is a remote queuing facility for WebSphere MQ for iSeries. It provides channel control programs for the WebSphere MQ for iSeries queue manager which form the interface to communication links, controllable by the system operator. The channel definitions held by distributed-queuing management use these communication links.

When a distributed-queuing management channel is started, it tries to use the connection specified in the channel definition. For this to succeed, it is necessary for the connection to be defined and available. This chapter explains how to do this.

Deciding on a connection

There are two forms of communication between WebSphere MQ for iSeries systems:

- iSeries TCP
  For TCP, a host address may be used, and these connections are set up as described in the OS/400 Communication Configuration Reference.
  In the TCP environment, each distributed service is allocated a unique TCP address which may be used by remote machines to access the service. The TCP address consists of a host name/number and a port number. All queue managers will use such a number to communicate with each other via TCP.

- iSeries SNA (LU 6.2)
  This form of communication requires the definition of an iSeries SNA logical unit type 6.2 (LU 6.2) that provides the physical link between the iSeries serving the local queue manager and the system serving the remote queue manager. Refer to the OS/400 Communication Configuration Reference for details on configuring communications in OS/400.

Defining a TCP connection

The channel definition contains a field, CONNECTION NAME, that contains either the TCP network address of the target, in dotted decimal form (for example 9.20.9.30) or the host name (for example AS4HUR1). If the CONNECTION NAME is a host name, a name server or the iSeries host table is used to convert the host name into a TCP host address.

A port number is required for a complete TCP address; if this is not supplied, the default port number 1414 is used. On the initiating end of a connection (sender, requester, and server channel types) it is possible to provide an optional port number for the connection, for example:

Connection name 9.20.9.30 (1555)

In this case the initiating end will attempt to connect to a receiving program at port 1555.
**Defining a TCP connection**

**Receiving on TCP**

Receiving channel programs are started in response to a startup request from the sending channel. To do this, a listener program has to be started to detect incoming network requests and start the associated channel. You start this listener program with the STRMQMLSR command.

You can start more than one listener for each queue manager. By default, the STRMQMLSR command uses port 1414 but you can override this. To override the default setting, add the following statements to the qm.ini file of the selected queue manager (in this example, the listener is required to use port 2500):

```
TCP:
  Port=2500
```

The qm.ini file is located in this IFS directory:

/QIBM/UserData/mqm/qmgrs/queue manager name.

This new value is read only when the TCP listener is started. If you have a listener already running, this change is not be seen by that program. To use the new value, stop the listener and issue the STRMQMLSR command again. Now, whenever you use the STRMQMLSR command, the listener defaults to the new port.

Alternatively, you can specify a different port number on the STRMQMLSR command. For example:

```
STRMQMLSR MQMNAME(queue manager name) PORT(2500)
```

This change makes the listener default to the new port for the duration of the listener job.

**Using the TCP SO_KEEPALIVE option**

If you want to use the SO_KEEPALIVE option (as discussed in “Checking that the other end of the channel is still available” on page 66) you must add the following entry to your queue manager configuration file (qm.ini in the IFS directory, /QIBM/UserData/mqm/qmgrs/queue manager name):

```
TCP:
  KeepAlive=yes
```

You must then issue the following command:

```
CFGTCP
```

Select option 3 (Change TCP Attributes). You can now specify a time interval in minutes. You can specify a value in the range 1 through 40320 minutes; the default is 120.

**Using the TCP listener backlog option**

When receiving on TCP, a maximum number of outstanding connection requests is set. This can be considered a backlog of requests waiting on the TCP port for the listener to accept the request.

The default listener backlog value on iSeries is 255. If the backlog reaches this value, the TCP connection is rejected and the channel will not be TCP: able to start.

For MCA channels, this results in the channel going into a RETRY state and retrying the connection at a later time.

For client connections, the client receives an MQRC_Q_MGR_NOT_AVAILABLE reason code from MQCONN and should retry the connection at a later time.
Defining a TCP connection

However, to avoid this error, you can add an entry in the qm.ini file:

```
ListenerBacklog = n
```

This overrides the default maximum number of outstanding requests (255) for the TCP listener.

Note: Some operating systems support a larger value than the default. If necessary, this can be used to avoid reaching the connection limit.

Defining an LU 6.2 connection

In WebSphere MQ for iSeries, a mode name, TP name, and connection name of a fully-qualified LU 6.2 connection can be used.

For other versions of WebSphere MQ for iSeries, a communications side information (CSI) object is required to define the LU 6.2 communications details for the sending end of a message channel. It is referred to in the CONNECTION NAME field of the Sender or Server channel definition for LU 6.2 connections. Further information on the communications side object is available in the AS/400 APPC Communications Programmer’s Guide.

The initiated end of the link must have a routing entry definition to complement this CSI object. Further information on managing work requests from remote LU 6.2 systems is available in the AS/400 Programming: Work Management Guide.

See the Multiplatform APPC Configuration Guide and the following table for information.

Table 47. Settings on the local OS/400 system for a remote queue manager platform

<table>
<thead>
<tr>
<th>Remote platform</th>
<th>TPNAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>z/OS, or OS/390 or MVS/ESA without CICS</td>
<td>The same as in the corresponding side information on the remote queue manager.</td>
</tr>
<tr>
<td>z/OS or OS/390 or MVS/ESA using CICS</td>
<td>CKRC relates to a sender channel on the OS/400 system. CKSV relates to a requester channel on the OS/400 system. CKRC relates to a server channel on the OS/400 system.</td>
</tr>
<tr>
<td>OS/400</td>
<td>The same as the compare value in the routing entry on the OS/400 system.</td>
</tr>
<tr>
<td>OS/2</td>
<td>As specified in the OS/2 Run Listener command, or defaulted from the OS/2 queue manager configuration file.</td>
</tr>
<tr>
<td>Compaq OpenVMS Alpha</td>
<td>As specified in the Digital OVMS Run Listener command.</td>
</tr>
<tr>
<td>Compaq NonStop Kernel</td>
<td>The same as the TPNAME specified in the receiver-channel definition.</td>
</tr>
<tr>
<td>Other UNIX systems</td>
<td>The invokable Transaction Program defined in the remote LU 6.2 configuration.</td>
</tr>
<tr>
<td>Windows</td>
<td>As specified in the Windows Run Listener command, or the invokable Transaction Program that was defined using TpSetup on Windows.</td>
</tr>
</tbody>
</table>

If you have more than one queue manager on the same machine, ensure that the TPnames in the channel definitions are unique.
### Defining an LU 6.2 connection

#### Initiating end (Sending)

Use the CRTMQMCHL command to define a channel of transport type *LU62*. For versions previous to WebSphere MQ for iSeries V5.3, define the name of the CSI object that this channel will use in the CONNECTION NAME field. (See "Creating a channel" on page 488 for details of how to do this.) Use of the CSI object is optional in WebSphere MQ for iSeries V5.1 or later.

The initiating end panel is shown in Figure 115. You press F10 from the first panel displayed to obtain the complete panel as shown.

![Create Comm Side Information (CRTCSI)](image)

**Figure 115. LU 6.2 communication setup panel - initiating end**

Complete the initiating end fields as follows:

**Side information**

Give this definition a name that will be used to store the side information object to be created, for example, WINSDOA1.

**Note**: For LU 6.2, the link between the message channel definition and the communication connection is the Connection name field of the message channel definition at the sending end. This field contains the name of the CSI object.

**Library**

The name of the library where this definition will be stored.

The CSI object must be available in a library accessible to the program serving the message channel, for example, QSYS, QMQM, and QGPL.

If the name is incorrect, missing, or cannot be found then an error will occur on channel start up.

**Remote location**

Specifies the remote location name with which your program communicates.
In short, this required parameter contains the logical unit name of the partner at the remote system, as defined in the device description that is used for the communication link between the two systems.

The **Remote location** name can be found by issuing the command DSPNETA on the remote system and seeing the default local location name.

**Transaction program**  
Specifies the name (up to 64 characters) of the transaction program on the remote system to be started. It may be a transaction process name, a program name, the channel name, or a character string that matches the **Compare value** in the routing entry.

This is a required parameter.

**Note:** To specify SNA service transaction program names, enter the hexadecimal representation of the service transaction program name. For example, to specify a service transaction program name whose hexadecimal representation is 21F0F0F1, you would enter X'21F0F0F1'.

More information on SNA service transaction program names is in the SNA Transaction Programmer’s Reference manual for LU Type 6.2.

If the receiving end is another iSeries system, the **Transaction program** name is used to match the CSI object at the sending end with the routing entry at the receiving end. This should be unique for each queue manager on the target iSeries system. (See the **Program to call** parameter under “Initiated end (Receiver)” on page 517) See also the **Comparison data: compare value** parameter in the Add Routing Entry panel.

**Text description**  
A description (up to 50 characters) to remind you of the intended use of this connection.

**Device**  
Specifies the name of the device description used for the remote system. The possible values are:

*LOC  
The device is determined by the system.

**Device-name**  
Specify the name of the device that is associated with the remote location.

**Local location**  
Specifies the local location name. The possible values are:

*LOC  
The local location name is determined by the system.

*NETATR  
The LCLLOCNAME value specified in the system network attributes is used.

**Local-location-name**  
Specify the name of your location. Specify the local location if you want to indicate a specific location name for the remote location. The location name can be found by using the DSPNETA command.
Defining an LU 6.2 connection

**Mode** Specifies the mode used to control the session. This name is the same as the Common Programming Interface (CPI)- Communications Mode_Name. The possible values are:

*NETATR
   The mode in the network attributes is used.

BLANK
   Eight blank characters are used.

**Mode-name**
   Specify a mode name for the remote location.

**Note:** Because the mode determines the transmission priority of the communications session, it may be useful to define different modes depending on the priority of the messages being sent; for example MQMODE_HI, MQMODE_MED, and MQMODE_LOW. (You can have more than one CSI pointing to the same location.)

**Remote network identifier**
   Specifies the remote network identifier used with the remote location. The possible values are:

*LOC
   The remote network ID for the remote location is used.

*NETATR
   The remote network identifier specified in the network attributes is used.

*NONE
   The remote network has no name.

**Remote-network-id**
   Specify a remote network ID. Use the DSPNETA command at the remote location to find the name of this network ID. It is the ‘local network ID’ at the remote location.

**Authority**
   Specifies the authority you are giving to users who do not have specific authority to the object, who are not on an authorization list, and whose group profile has no specific authority to the object. The possible values are:

*LIBCRTAUT
   Public authority for the object is taken from the CRTAUT parameter of the specified library. This value is determined at create time. If the CRTAUT value for the library changes after the object is created, the new value does not affect existing objects.

*CHANGE
   Change authority allows the user to perform basic functions on the object, however, the user cannot change the object. Change authority provides object operational authority and all data authority.

*ALL
   The user can perform all operations except those limited to the owner or controlled by authorization list management authority. The user can control the object’s existence and specify the security for the object, change the object, and perform basic functions on the object. The user can change ownership of the object.
**Defining an LU 6.2 connection**

**USE** Use authority provides object operational authority and read authority.

**EXCLUDE** Exclude authority prevents the user from accessing the object.

**Authorization-list**
Specify the name of the authorization list whose authority is used for the side information.

**Initiated end (Receiver)**

Use the CRTMQMCHL command to define the receiving end of the message channel link with transport type *LU62. Leave the CONNECTION NAME field blank and ensure that the corresponding details match the sending end of the channel. (See "Creating a channel" on page 488 for details of how to do this.)

To enable the initiating end to start the receiving channel, add a routing entry to a subsystem at the initiated end. The subsystem must be the one that allocates the APPC device used in the LU 6.2 sessions and, therefore, it must have a valid communications entry for that device. The routing entry calls the program that starts the receiving end of the message channel.

Use the OS/400 commands (for example, ADDRTGE) to define the end of the link that is initiated by a communication session.

The initiated end panel is shown in Figure [Figure 116].

---

### Add Routing Entry (ADDRTGE)

Type choices, press Enter.

- **Subsystem description** ......... QCMN Name
- **Library** ........... *LIBL Name, +LIBL, +CURLIB
- **Routing entry sequence number** .. 1 1-9999
- **Comparison data:**
  - **Compare value** ........ MQSERIES
  - **Starting position** ......... 37 1-80
  - **Program to call** ............ AMQCR6B Name, +RTGDATA
  - **Class** ............. *SBSD Name, +SBSD
  - **Library** ........... +LIBL Name, +LIBL, +CURLIB
  - **Maximum active routing steps** .. +NOMAX 0-1000, +NOMAX
  - **Storage pool identifier** ....... 1 1-10

---

**Figure 116. LU 6.2 communication setup panel - initiated end**

**Subsystem description**

The name of your subsystem where this definition resides. Use the OS/400 WRKSBSD command to view and update the appropriate subsystem description for the routing entry.
Defining an LU 6.2 connection

Routing entry sequence number
A unique number in your subsystem to identify this communication definition. You can use values in the range 1 to 9999.

Comparison data: Compare value
A text string to compare with that received when the session is started by a Transaction program parameter, as shown in Figure 115 on page 514. The character string is derived from the Transaction program field of the sender CSI.

Comparison data: Starting position
The character position in the string where the comparison is to start.

Note: The starting position field is the character position in the string for comparison, and this is always 37.

Program to call
The name of the program that runs the inbound message program to be called to start the session.

The program, AMQCRC6A, is called for the default queue manager. This is a program supplied with WebSphere MQ for iSeries that sets up the environment and then calls AMQCRS6A.

For additional queue managers:
• Each queue manager has a specific LU 6.2 invokable program located in its library. This program is called AMQCRC6B and is automatically generated when the queue manager is created.
• Each queue manager requires a specific routing entry with unique routing data to be added. This routing data should match the Transaction program name supplied by the requesting system (see “Initiating end (Sending)” on page 514).

An example of this is shown in Figure 117:

```
Display Routing Entries
System: MY400

Subsystem description: QCMN Status: ACTIVE

Type options, press Enter.
5=Display details

<table>
<thead>
<tr>
<th>Opt</th>
<th>Seq Nbr</th>
<th>Program</th>
<th>Library</th>
<th>Compare Value</th>
<th>Start Pos</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td></td>
<td>*RTGDTA</td>
<td>'QZCSRSVR'</td>
<td>37</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td></td>
<td>*RTGDTA</td>
<td>'QZRCRSVR'</td>
<td>37</td>
<td></td>
</tr>
<tr>
<td>30</td>
<td></td>
<td>*RTGDTA</td>
<td>'QZHQTRG'</td>
<td>37</td>
<td></td>
</tr>
<tr>
<td>50</td>
<td></td>
<td>*RTGDTA</td>
<td>'QVPRINT'</td>
<td>37</td>
<td></td>
</tr>
<tr>
<td>60</td>
<td></td>
<td>*RTGDTA</td>
<td>'QNPSESSION'</td>
<td>37</td>
<td></td>
</tr>
<tr>
<td>70</td>
<td></td>
<td>*RTGDTA</td>
<td>'QNPSESSION'</td>
<td>37</td>
<td></td>
</tr>
<tr>
<td>80</td>
<td></td>
<td>QNMAREXEC</td>
<td>QSYS'</td>
<td>'AREXEC'</td>
<td>37</td>
</tr>
<tr>
<td>90</td>
<td></td>
<td>AMQCRC6A</td>
<td>QMQMBW</td>
<td>'QMSESERS'</td>
<td>37</td>
</tr>
<tr>
<td>100</td>
<td></td>
<td>*RTGDTA</td>
<td>'QTFDWNL'</td>
<td>37</td>
<td></td>
</tr>
<tr>
<td>150</td>
<td></td>
<td>*RTGDTA</td>
<td>'QMFRCVR'</td>
<td>37</td>
<td></td>
</tr>
</tbody>
</table>

F3=Exit   F9=Display all detailed descriptions   F12=Cancel
```

Figure 117. LU 6.2 communication setup panel - initiated end
In Figure 117 sequence number 90 represents the default queue manager and provides compatibility with configurations from previous releases (that is, V3R2, V3R6, V3R7, and V4R2) of WebSphere MQ for iSeries. These releases allow one queue manager only. Sequence numbers 92 and 94 represent two additional queue managers called ALPHA and BETA that are created with libraries QMALPHA and QMBETA.

**Note:** You can have more than one routing entry for each queue manager by using different routing data. This gives the option of different job priorities depending on the classes used.

**Class**
The name and library of the class used for the steps started through this routing entry. The class defines the attributes of the routing step’s running environment and specifies the job priority. An appropriate class entry must be specified. Use, for example, the WRKCLS command to display existing classes or to create a new class. Further information on managing work requests from remote LU 6.2 systems is available in the *AS/400 Programming: Work Management Guide*.

**Note on Work Management**
The AMQCRS6A job will not be able to take advantage of the normal iSeries work management features that are documented in the *WebSphere MQ for iSeries V5.3 System Administration* book because it is not started in the same way as other WebSphere MQ jobs. To change the run-time properties of the LU62 receiver jobs, you can do one of the following:

- Alter the class description that is specified on the routing entry for the AMQCRS6A job
- Change the job description on the communications entry

See the *AS/400 Programming: Work Management Guide* for more information about configuring Communication Jobs.
DQM in WebSphere MQ for iSeries
Chapter 42. Example configuration - IBM WebSphere MQ for iSeries

This chapter gives an example of how to set up communication links from WebSphere MQ for iSeries to WebSphere MQ products on the following platforms:

- OS/2
- Windows
- AIX
- Compaq Tru64 UNIX
- HP-UX
- AT&T GIS UNIX\(^9\)
- Solaris
- z/OS, OS/390, or MVS/ESA without CICS
- VSE/ESA

First it describes the parameters needed for an LU 6.2 connection, then it describes “Establishing an LU 6.2 connection” on page 526 and “Establishing a TCP connection” on page 532.

Once the connection is established, you need to define some channels to complete the configuration. This is described in “WebSphere MQ for iSeries configuration” on page 534.

See Chapter 7, “Example configuration chapters in this book” on page 101 for background information about this chapter and how to use it.

### Configuration parameters for an LU 6.2 connection

Table 48 presents a worksheet listing all the parameters needed to set up communication from iSeries to one of the other WebSphere MQ platforms. The worksheet shows examples of the parameters, which have been tested in a working environment, and leaves space for you to fill in your own values. An explanation of the parameter names follows the worksheet. Use the worksheet in this chapter in conjunction with the worksheet in the chapter for the platform to which you are connecting.

#### Configuration worksheet

Use the following worksheet to record the values you will use for this configuration. Where numbers appear in the Reference column they indicate that the value must match that in the appropriate worksheet elsewhere in this book. The examples that follow in this chapter refer back to the values in the ID column of this table. The entries in the Parameter Name column are explained in “Explanation of terms” on page 524.

<table>
<thead>
<tr>
<th>ID</th>
<th>Parameter Name</th>
<th>Reference</th>
<th>Example Used</th>
<th>User Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Local network ID</td>
<td></td>
<td>NETID</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Local control point name</td>
<td></td>
<td>AS400PU</td>
<td></td>
</tr>
</tbody>
</table>

9. This platform has become NCR UNIX SVR4 MP-RAS, R3.0

© Copyright IBM Corp. 1993, 2002
### Table 48. Configuration worksheet for SNA on an iSeries system (continued)

<table>
<thead>
<tr>
<th>ID</th>
<th>Parameter Name</th>
<th>Reference</th>
<th>Example Used</th>
<th>User Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>LU name</td>
<td>AS400LU</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>LAN destination address</td>
<td>10005A5962EF</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Subsystem description</td>
<td>QCMN</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>Line description</td>
<td>TOKENRINGL</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>Resource name</td>
<td>LIN041</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>Local Transaction Program name</td>
<td>MQSERIES</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Connection to an OS/2 system**

The values in this section must match those used in Table 15 on page 144, as indicated.

<table>
<thead>
<tr>
<th>ID</th>
<th>Parameter Name</th>
<th>Reference</th>
<th>Example Used</th>
<th>User Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>9</td>
<td>Network ID</td>
<td>NETID</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>Control point name</td>
<td>OS2PU</td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>LU name</td>
<td>OS2LU</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>Controller description</td>
<td>OS2PU</td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>Device</td>
<td>OS2LU</td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>Side information</td>
<td>OS2CPIC</td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>Transaction Program</td>
<td>MQSERIES</td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>LAN adapter address</td>
<td>10005AFC5D83</td>
<td></td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>Mode</td>
<td>#INTER</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Connection to a Windows system**

The values in this section must match those used in Table 17 on page 168, as indicated.

<table>
<thead>
<tr>
<th>ID</th>
<th>Parameter Name</th>
<th>Reference</th>
<th>Example Used</th>
<th>User Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>9</td>
<td>Network ID</td>
<td>NETID</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>Control point name</td>
<td>WINNTCP</td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>LU name</td>
<td>WINNTLU</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>Controller description</td>
<td>WINNTCP</td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>Device</td>
<td>WINNTLU</td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>Side information</td>
<td>NTCPIC</td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>Transaction Program</td>
<td>MQSERIES</td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>LAN adapter address</td>
<td>08005AA5FAB9</td>
<td></td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>Mode</td>
<td>#INTER</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Connection to an AIX system**

The values in this section must match those used in Table 21 on page 197, as indicated.

<table>
<thead>
<tr>
<th>ID</th>
<th>Parameter Name</th>
<th>Reference</th>
<th>Example Used</th>
<th>User Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>9</td>
<td>Network ID</td>
<td>NETID</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>Control point name</td>
<td>AIXPU</td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>LU name</td>
<td>AIXLU</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>Controller description</td>
<td>AIXPU</td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>Device</td>
<td>AIXLU</td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>Side information</td>
<td>AIXCPIC</td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>Transaction Program</td>
<td>MQSERIES</td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>LAN adapter address</td>
<td>123456789012</td>
<td></td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>Mode</td>
<td>#INTER</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Connection to an HP-UX system**

The values in this section must match those used in Table 24 on page 219, as indicated.

<table>
<thead>
<tr>
<th>ID</th>
<th>Parameter Name</th>
<th>Reference</th>
<th>Example Used</th>
<th>User Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>9</td>
<td>Network ID</td>
<td>NETID</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>Control point name</td>
<td>HPUXPU</td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>LU name</td>
<td>HPUXLU</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Table 48. Configuration worksheet for SNA on an iSeries system (continued)

<table>
<thead>
<tr>
<th>ID</th>
<th>Parameter Name</th>
<th>Reference</th>
<th>Example Used</th>
<th>User Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>12</td>
<td>Controller description</td>
<td></td>
<td>HPUXPU</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>Device</td>
<td></td>
<td>HPUXLU</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>Side information</td>
<td></td>
<td>HPUXCPIC</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>Transaction Program</td>
<td>7</td>
<td>MQSERIES</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>LAN adapter address</td>
<td>8</td>
<td>100090DC2C7C</td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>Mode</td>
<td>17</td>
<td>#INTER</td>
<td></td>
</tr>
</tbody>
</table>

**Connection to an AT&T GIS UNIX system**

The values in this section must match those used in [Table 26 on page 243](#) as indicated.

<table>
<thead>
<tr>
<th>ID</th>
<th>Parameter Name</th>
<th>Reference</th>
<th>Example Used</th>
<th>User Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>9</td>
<td>Network ID</td>
<td>2</td>
<td>NETID</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>Control point name</td>
<td>3</td>
<td>GISPU</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>LU name</td>
<td>4</td>
<td>GISLU</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>Controller description</td>
<td></td>
<td>GISPU</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>Device</td>
<td></td>
<td>GISLU</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>Side information</td>
<td></td>
<td>GISCPI</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>Transaction Program</td>
<td>5</td>
<td>MQSERIES</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>LAN adapter address</td>
<td>8</td>
<td>10007038E86B</td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>Mode</td>
<td>15</td>
<td>#INTER</td>
<td></td>
</tr>
</tbody>
</table>

**Connection to a Solaris system**

The values in this section must match those used in [Table 28 on page 258](#) as indicated.

<table>
<thead>
<tr>
<th>ID</th>
<th>Parameter Name</th>
<th>Reference</th>
<th>Example Used</th>
<th>User Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>9</td>
<td>Network ID</td>
<td>2</td>
<td>NETID</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>Control point name</td>
<td>3</td>
<td>SOLARPU</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>LU name</td>
<td>7</td>
<td>SOLARLU</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>Controller description</td>
<td></td>
<td>SOLARPU</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>Device</td>
<td></td>
<td>SOLARLU</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>Side information</td>
<td></td>
<td>SOLCPIC</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>Transaction Program</td>
<td>8</td>
<td>MQSERIES</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>LAN adapter address</td>
<td>5</td>
<td>08002071CC8A</td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>Mode</td>
<td>17</td>
<td>#INTER</td>
<td></td>
</tr>
</tbody>
</table>

**Connection to a z/OS system without CICS**

The values in this section must match those used in [Table 33 on page 366](#) as indicated.

<table>
<thead>
<tr>
<th>ID</th>
<th>Parameter Name</th>
<th>Reference</th>
<th>Example Used</th>
<th>User Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>9</td>
<td>Network ID</td>
<td>2</td>
<td>NETID</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>Control point name</td>
<td>3</td>
<td>MVSPU</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>LU name</td>
<td>4</td>
<td>MVSLU</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>Controller description</td>
<td></td>
<td>MVSPU</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>Device</td>
<td></td>
<td>MVSLU</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>Side information</td>
<td></td>
<td>MVSCPIC</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>Transaction Program</td>
<td>7</td>
<td>MQSERIES</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>LAN adapter address</td>
<td>8</td>
<td>400074511092</td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>Mode</td>
<td>6</td>
<td>#INTER</td>
<td></td>
</tr>
</tbody>
</table>

**Connection to a VSE/ESA system**

The values in this section must match those used in [Table 50 on page 549](#) as indicated.

<table>
<thead>
<tr>
<th>ID</th>
<th>Parameter Name</th>
<th>Reference</th>
<th>Example Used</th>
<th>User Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>9</td>
<td>Network ID</td>
<td>1</td>
<td>NETID</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>Control point name</td>
<td>2</td>
<td>VSEPU</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>LU name</td>
<td>3</td>
<td>VSELU</td>
<td></td>
</tr>
</tbody>
</table>
OS/400 and LU 6.2

Table 48. Configuration worksheet for SNA on an iSeries system (continued)

<table>
<thead>
<tr>
<th>ID</th>
<th>Parameter Name</th>
<th>Reference</th>
<th>Example Used</th>
<th>User Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>12</td>
<td>Controller description</td>
<td>VSEPU</td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>Device</td>
<td>VSELU</td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>Side information</td>
<td>VSECPIC</td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>Transaction Program</td>
<td>MQ01</td>
<td>MQ01</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>LAN adapter address</td>
<td>400074511092</td>
<td></td>
<td>MQ01</td>
</tr>
<tr>
<td>17</td>
<td>Mode</td>
<td>#INTER</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Explanation of terms

1. See “How to find network attributes” on page 525 for the details of how to find the configured values.

4. **LAN destination address**
   The hardware address of the iSeries system token-ring adapter. You can find the value using the command DSPLIND Line description (6).

5. **Subsystem description**
   This is the name of any OS/400 subsystem that will be active while using the queue manager. The name QCMN has been used because this is the OS/400 communications subsystem.

6. **Line description**
   If this has been specified it is indicated in the Description field of the resource Resource name. See “How to find the value of Resource name” on page 525 for details. If the value is not specified you will need to create a line description.

7. **Resource name**
   See “How to find the value of Resource name” on page 525 for details of how to find the configured value.

8. **Local Transaction Program name**
   WebSphere MQ applications trying to converse with this workstation will specify a symbolic name for the program to be run at the receiving end. This will have been defined on the channel definition at the sender. For simplicity, wherever possible use a transaction program name of MQSERIES, or in the case of a connection to VSE/ESA, where the length is limited to 4 bytes, use MQTP.
   See Table 47 on page 513 for more information.

12. **Controller description**
   This is an alias for the Control Point name (or Node name) of the partner system. For convenience we have used the actual name of the partner in this example.

13. **Device**
   This is an alias for the LU of the partner system. For convenience we have used the LU name of the partner in this example.

14. **Side information**
   This is the name given to the CPI-C side information profile. You specify your own 8-character name for this.
How to find network attributes
The local node has been partially configured as part of the OS/400 installation. To display the current network attributes enter the command DSPNETA.

If you need to change these values use the command CHGNETA. An IPL may be required to apply your changes.

<table>
<thead>
<tr>
<th>Display Network Attributes</th>
<th>System: AS400PU</th>
</tr>
</thead>
<tbody>
<tr>
<td>Current system name</td>
<td>AS400PU</td>
</tr>
<tr>
<td>Pending system name</td>
<td></td>
</tr>
<tr>
<td>Local network ID</td>
<td>NETID</td>
</tr>
<tr>
<td>Local control point name</td>
<td>AS400PU</td>
</tr>
<tr>
<td>Default local location</td>
<td>AS400LU</td>
</tr>
<tr>
<td>Default mode</td>
<td>BLANK</td>
</tr>
<tr>
<td>APPN node type</td>
<td>*ENDNODE</td>
</tr>
<tr>
<td>Data compression</td>
<td>*NONE</td>
</tr>
<tr>
<td>Intermediate data compression</td>
<td>*NONE</td>
</tr>
<tr>
<td>Maximum number of intermediate sessions</td>
<td>200</td>
</tr>
<tr>
<td>Route addition resistance</td>
<td>128</td>
</tr>
<tr>
<td>Server network ID/control point name</td>
<td>NETID NETCP</td>
</tr>
</tbody>
</table>

Press Enter to continue.
F3=Exit  F12=Cancel

Check that the values for Local network ID (1), Local control point name (2), and Default local location (3), correspond to the values on your worksheet.

How to find the value of Resource name
Type WRKHDWRSC TYPE(*CMN) and press Enter. The Work with Communication Resources panel is displayed. The value for Resource name is found as the Token-Ring Port. It is LIN041 in this example.
Establishing an LU 6.2 connection

This section describes how to establish an LU 6.2 connection.

Local node configuration

To configure the local node, you need to:
1. Create a line description
2. Add a routing entry

Creating a line description

1. If the line description has not already been created use the command CRTLINTRN.
2. Specify values for Line description (6) and Resource name (7).
Adding a routing entry

1. Type the command ADDRTGE and press Enter.

2. Specify your value for **Subsystem description** (5), and the values shown here for **Routing entry sequence number**, **Compare value**, **Starting position**, **Program to call**, and the **Library** containing the program to call.

3. Type the command STRSBS * subsystem description (5) and press Enter.
Connection to partner node

This example is for a connection to an OS/2 system, but the steps are the same for other nodes. The steps are:
1. Create a controller description.
2. Create a device description.
3. Create CPI-C side information.
4. Add a communications entry for APPC.
5. Add a configuration list entry.

Creating a controller description
1. At a command line type CRTCTLAPPC and press Enter.

2. Specify a value for Controller description (12), set Link type to *LAN, and set Online at IPL to *NO.
3. Press Enter twice, followed by F10.
4. Specify values for **Switched line list** (6), **Remote network identifier** (9), **Remote control point** (10), and **LAN remote adapter address** (16).

5. Press Enter.

**Creating a device description**

1. Type the command CRTDEVAPPC and press Enter.

---

### CreateCtl Desc (APPC) (CRTCTLAPPC)

Type choices, press Enter.

- **Controller description**... > OS2PU Name
- **Link type**......... > *LAN *FAX, *FR, *IDLC, *LAN...
- **Online at IPL**..... > *NO +YES, *NO
- **APPN-capable**..... > *YES +YES, *NO
- **Switched line list**... TOKERNING Name
  + for more values
- **Maximum frame size**..... > 265-16393, 256, 265, 512...
- **Remote control point**... OS2PU Name, *ANY
- **Exchange identifier**.... > OS2PU Name, *ANY
- **Initial connection**....... > *DIAL *DIAL, *ANS
- **Dial initiation**....... > *LINKTYPE *LINKTYPE, *IMMED, *DELAY
- **LAN remote adapter address**... 1000SAFC5D83 000000000001-FFFFFFFFFFFF
- **APPN CP session support**... *YES +YES, *NO
- **APPN node type**....... > *ENDNODE +ENDNODE, *LENODE...
- **APPN transmission group number**... 1 1-20, *CALC

More...

F3=Exit F4=Prompt F5=Refresh F12=Cancel F13=How to use this display
F24=More keys

---

### Create Device Desc (APPC) (CRTDEVAPPC)

Type choices, press Enter.

- **Device description**... OS2LU Name
- **Remote location**...... OS2LU Name
- **Online at IPL**....... > *YES +YES, *NO
- **Local location**...... AS400LU Name, *NETATR
- **Remote network identifier**... NETID Name, *NETATR, *NONE
- **Attached controller**... OS2PU Name
- **Mode**.............. > *NETATR +NETATR Name, *NETATR
  + for more values
- **Message queue**.... > QSYSOPR Name, QSYSOPR
- **Library**........... *LIBL Name, *LIBL, *CURLIB
- **APPN-capable**..... > *YES +YES, *NO
- **Single session:**
  - **Single session capable**... > *NO +NO, *YES
  - **Number of conversations**... 1-512

More...

F3=Exit F4=Prompt F5=Refresh F12=Cancel F13=How to use this display
F24=More keys

Parameter DEVD required. +

**Note:** You can avoid having to create controller and device descriptions manually by taking advantage of OS/400's auto-configuration service. Consult the OS/400 documentation for details.

### Creating CPI-C side information

1. Type **CRTCSI** and press F10.

```
Create Comm Side Information (CRTCSI)

Type choices, press Enter.

Side information ........ OS2CPIC Name
Library ................. *CURLIB Name, *CURLIB
Remote location ........ OS2LU Name
Transaction program .... MQSERIES
Text 'description' ....... *BLANK
```

**Additional Parameters**

- **Device** ............. *LOC Name, *LOC
- **Local location** ...... AS400LU Name, *LOC, *NETATR
- **Mode** ............... #INTER Name, *NETATR
- **Authority** .......... *LIBCRTAUT Name, *LIBCRTAUT, *CHANGE...

**Parameter CSI required.**

```
F3=Exit   F4=Prompt   F5=Refresh   F12=Cancel   F13=How to use this display
F24=More keys
```

2. Specify values for **Side information**, **Remote location**, **Transaction program**, **Local location**, **Mode**, and **Remote network identifier**.

3. Press Enter.

### Adding a communications entry for APPC

1. At a command line type **ADDCMNE** and press Enter.
2. Specify values for **Subsystem description** and **Device**, and press Enter.

**Adding a configuration list entry**

1. Type ADDCFGLE *APPNRMT* and press F4.

2. Specify values for **Remote location name**, **Remote network identifier**, **Local location name**, **Remote control point**, and **Control point net ID**.
3. Press Enter.

What next?
The LU 6.2 connection is now established. You are ready to complete the
configuration. Go to “WebSphere MQ for iSeries configuration” on page 534

Establishing a TCP connection
If TCP is already configured there are no extra configuration tasks. The following
panels guide you through the steps that may be required if TCP/IP is not
configured.

Adding a TCP/IP interface
1. At a command line type ADDTCPIFC and press Enter.

   Add TCP/IP Interface (ADDTCPIFC)
   Type choices, press Enter.
   Internet address ........ 19.22.11.55
   Line description .......... TOKENRINGL Name, *LOOPBACK
   Subnet mask ............... 255.255.0.0
   Type of service .......... +NORMAL *MINDELAY, *MATHPRUT..
   Maximum transmission unit .... +LIND 576-16388, +LIND
   Autostart ................. +YES *YES, +NO
   PVC logical channel identifier ... 001-FFF
   + for more values
   X.25 idle circuit timeout ... 60 1-600
   X.25 maximum virtual circuits . 64 0-64
   X.25 DDN interface ......... +NO *YES, +NO
   TRLAN bit sequencing ....... +MSB *MSB, +LSB

   F3=Exit  F4=Prompt  F5=Refresh  F12=Cancel  F13=How to use this display
   F24=More keys

2. Specify this machine’s Internet address and Line description, and a Subnet
   mask.
3. Press Enter.

Adding a TCP/IP loopback interface
1. At a command line type ADDTCPIFC and press Enter.
2. Specify the values for **Internet address**, **Line description**, and **Subnet mask**.

**Adding a default route**

1. At a command line type `ADDTCPRTE` and press Enter.

2. Fill in with values appropriate to your network and press Enter to create a default route entry.
OS/400 and TCP

What next?

The TCP connection is now established. You are ready to complete the configuration. Go to “WebSphere MQ for iSeries configuration”.

WebSphere MQ for iSeries configuration

Start the TCP channel listener using the command STRMQMLSR.

Start any sender channel using the command STRMQMCHL
CHLNAME(channel_name).

Use the WRKMQMQ command to display the WebSphere MQ configuration menu.

Note: AMQ* errors are placed in the log relating to the job that found the error. Use the WRKACTJOB command to display the list of jobs. Under the subsystem name QSYSWRK, locate the job and enter 5 against it to work with that job. WebSphere MQ logs are prefixed ‘AMQ’.

Basic configuration

1. First you need to create a queue manager. To do this, type CRTMQM and press Enter.

   Create Message Queue Manager (CRTMQM)

   Type choices, press Enter.
   Message Queue Manager name ...
   Text 'description' ......... *BLANK
   Trigger interval ............ 999999999 0-999999999
   Undelivered message queue .... *NONE
   Default transmission queue ... *NONE
   Maximum handle limit ....... 256 1-999999999
   Maximum uncommitted messages .. 1000 1-10000
   Default Queue manager ....... *NO *YES, *NO

   Bottom
   F3=Exit  F4=Prompt  F5=Refresh  F12=Cancel  F13=How to use this display
   F24=More keys

2. In the Message Queue Manager name field, type AS400. In the Undelivered message queue field, type DEAD.LETTER.QUEUE.

3. Press Enter.

4. Now start the queue manager by entering STRMQM MQMNAME(AS400).

5. Create the undelivered message queue using the following parameters. (For details and an example refer to “Defining a queue” on page 538)

   Local Queue
   Queue name : DEAD.LETTER.QUEUE
   Queue type : *LCL
Channel configuration

This section details the configuration to be performed on the OS/400 queue manager to implement the channel described in Figure 32 on page 101.

Examples are given for connecting WebSphere MQ for iSeries and MQSeries for OS/2 Warp. If you wish connect to another WebSphere MQ product, use the appropriate values from the table in place of those for OS/2.

Notes:
1. The words in **bold** are user-specified and reflect the names of WebSphere MQ objects used throughout these examples. If you change the names used here, ensure that you also change the other references made to these objects throughout this book. All others are keywords and should be entered as shown.
2. The WebSphere MQ channel ping command (PNGMQMCHL) runs interactively, whereas starting a channel causes a batch job to be submitted. If a channel ping completes successfully but the channel will not start, this indicates that the network and WebSphere MQ definitions are probably correct, but that the OS/400 environment for the batch job is not. For example, make sure that QSYS2 is included in the system portion of the library list and not just your personal library list.

For details and examples of how to create the objects listed refer to “Defining a queue” on page 538 and “Defining a channel” on page 539.

### Table 49. Configuration worksheet for WebSphere MQ for iSeries

<table>
<thead>
<tr>
<th>ID</th>
<th>Parameter Name</th>
<th>Reference</th>
<th>Example Used</th>
<th>User Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>Queue Manager Name</td>
<td></td>
<td>AS400</td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>Local queue name</td>
<td></td>
<td>AS400.LOCALQ</td>
<td></td>
</tr>
</tbody>
</table>

**Definition for local node**

**Connection to MQSeries for OS/2 Warp**

The values in this section of the table must match those used in Table 16 on page 161 as indicated.

| C  | Remote queue manager name | A | OS2 |
| D  | Remote queue name         |   | OS2.REMOTEQ |
| E  | Queue name at remote system | B | OS2.LOCALQ |
| F  | Transmission queue name   |   | OS2 |
| G  | Sender (SNA) channel name |   | AS400.OS2.SNA |
| H  | Sender (TCP) channel name |   | AS400.OS2.TCP |
| I  | Receiver (SNA) channel name | G | OS2.AS400.SNA |
| J  | Receiver (TCP) channel name | H | OS2.AS400.TCP |

**Connection to WebSphere MQ for Windows**

The values in this section of the table must match those used in Table 18 on page 184 as indicated.

| C  | Remote queue manager name | A | WINNT |
| D  | Remote queue name         |   | WINNT.REMOTEQ |
| E  | Queue name at remote system | B | WINNT.LOCALQ |
| F  | Transmission queue name   |   | WINNT |
| G  | Sender (SNA) channel name |   | AS400.WINNT.SNA |
| H  | Sender (TCP/IP) channel name |   | AS400.WINNT.TCP |
| I  | Receiver (SNA) channel name | G | WINNT.AS400.SNA |
| J  | Receiver (TCP/IP) channel name | H | WINNT.AS400.TCP |
### OS/400 configuration

**Table 49. Configuration worksheet for WebSphere MQ for iSeries (continued)**

<table>
<thead>
<tr>
<th>ID</th>
<th>Parameter Name</th>
<th>Reference</th>
<th>Example Used</th>
<th>User Value</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><strong>Connection to WebSphere MQ for AIX</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>Remote queue manager name</td>
<td>AIX</td>
<td></td>
<td></td>
</tr>
<tr>
<td>D</td>
<td>Remote queue name</td>
<td>AIX.REMOTEQ</td>
<td></td>
<td></td>
</tr>
<tr>
<td>E</td>
<td>Queue name at remote system</td>
<td>AIX.LOCALQ</td>
<td></td>
<td></td>
</tr>
<tr>
<td>F</td>
<td>Transmission queue name</td>
<td>AIX</td>
<td></td>
<td></td>
</tr>
<tr>
<td>G</td>
<td>Sender (SNA) channel name</td>
<td>AS400.AIX.SNA</td>
<td></td>
<td></td>
</tr>
<tr>
<td>H</td>
<td>Sender (TCP/IP) channel name</td>
<td>AS400.AIX.TCP</td>
<td></td>
<td></td>
</tr>
<tr>
<td>I</td>
<td>Receiver (SNA) channel name</td>
<td>AIX.AS400.SNA</td>
<td></td>
<td></td>
</tr>
<tr>
<td>J</td>
<td>Receiver (TCP) channel name</td>
<td>AS400.AIX.TCP</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Connection to MQSeries for Compaq Tru64 UNIX</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>Remote queue manager name</td>
<td>DECUX</td>
<td></td>
<td></td>
</tr>
<tr>
<td>D</td>
<td>Remote queue name</td>
<td>DECUX.REMOTEQ</td>
<td></td>
<td></td>
</tr>
<tr>
<td>E</td>
<td>Queue name at remote system</td>
<td>DECUX.LOCALQ</td>
<td></td>
<td></td>
</tr>
<tr>
<td>F</td>
<td>Transmission queue name</td>
<td>DECUX</td>
<td></td>
<td></td>
</tr>
<tr>
<td>G</td>
<td>Sender (TCP) channel name</td>
<td>DECUX.AS400.TCP</td>
<td></td>
<td></td>
</tr>
<tr>
<td>J</td>
<td>Receiver (TCP) channel name</td>
<td>AS400.DECUX.TCP</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Connection to WebSphere MQ for HP-UX</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>Remote queue manager name</td>
<td>HPUX</td>
<td></td>
<td></td>
</tr>
<tr>
<td>D</td>
<td>Remote queue name</td>
<td>HPUX.REMOTEQ</td>
<td></td>
<td></td>
</tr>
<tr>
<td>E</td>
<td>Queue name at remote system</td>
<td>HPUX.LOCALQ</td>
<td></td>
<td></td>
</tr>
<tr>
<td>F</td>
<td>Transmission queue name</td>
<td>HPUX</td>
<td></td>
<td></td>
</tr>
<tr>
<td>G</td>
<td>Sender (SNA) channel name</td>
<td>AS400.HPUX.SNA</td>
<td></td>
<td></td>
</tr>
<tr>
<td>H</td>
<td>Sender (TCP) channel name</td>
<td>AS400.HPUX.TCP</td>
<td></td>
<td></td>
</tr>
<tr>
<td>I</td>
<td>Receiver (SNA) channel name</td>
<td>HPUX.AS400.SNA</td>
<td></td>
<td></td>
</tr>
<tr>
<td>J</td>
<td>Receiver (TCP) channel name</td>
<td>HPUX.AS400.TCP</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Connection to MQSeries for AT&amp;T GIS UNIX</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>Remote queue manager name</td>
<td>GIS</td>
<td></td>
<td></td>
</tr>
<tr>
<td>D</td>
<td>Remote queue name</td>
<td>GIS.REMOTEQ</td>
<td></td>
<td></td>
</tr>
<tr>
<td>E</td>
<td>Queue name at remote system</td>
<td>GIS.LOCALQ</td>
<td></td>
<td></td>
</tr>
<tr>
<td>F</td>
<td>Transmission queue name</td>
<td>GIS</td>
<td></td>
<td></td>
</tr>
<tr>
<td>G</td>
<td>Sender (SNA) channel name</td>
<td>AS400.GIS.SNA</td>
<td></td>
<td></td>
</tr>
<tr>
<td>H</td>
<td>Sender (TCP) channel name</td>
<td>AS400.GIS.TCP</td>
<td></td>
<td></td>
</tr>
<tr>
<td>I</td>
<td>Receiver (SNA) channel name</td>
<td>GIS.AS400.SNA</td>
<td></td>
<td></td>
</tr>
<tr>
<td>J</td>
<td>Receiver (TCP/IP) channel name</td>
<td>GIS.AS400.TCP</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Connection to WebSphere MQ for Solaris</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>Remote queue manager name</td>
<td>SOLARIS</td>
<td></td>
<td></td>
</tr>
<tr>
<td>D</td>
<td>Remote queue name</td>
<td>SOLARIS.REMOTEQ</td>
<td></td>
<td></td>
</tr>
<tr>
<td>E</td>
<td>Queue name at remote system</td>
<td>SOLARIS.LOCALQ</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Table 49. Configuration worksheet for WebSphere MQ for iSeries (continued)

<table>
<thead>
<tr>
<th>ID</th>
<th>Parameter Name</th>
<th>Reference</th>
<th>Example Used</th>
<th>User Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>F</td>
<td>Transmission queue name</td>
<td>SOLARIS</td>
<td></td>
<td></td>
</tr>
<tr>
<td>G</td>
<td>Sender (SNA) channel name</td>
<td>AS400.SOLARIS.SNA</td>
<td></td>
<td></td>
</tr>
<tr>
<td>H</td>
<td>Sender (TCP/IP) channel name</td>
<td>AS400.SOLARIS.TCP</td>
<td></td>
<td></td>
</tr>
<tr>
<td>I</td>
<td>Receiver (SNA) channel name</td>
<td>SOLARIS.AS400.SNA</td>
<td></td>
<td></td>
</tr>
<tr>
<td>J</td>
<td>Receiver (TCP/IP) channel name</td>
<td>SOLARIS.AS400.TCP</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Connection to WebSphere MQ for z/OS without CICS

The values in this section of the table must match those used in Table 34 on page 372 as indicated.

| C  | Remote queue manager name | MVS |
| D  | Remote queue name         | MVS.REMOTEQ |
| E  | Queue name at remote system | MVS.LOCALQ |
| F  | Transmission queue name   | MVS |
| G  | Sender (SNA) channel name | AS400.MVS.SNA |
| H  | Sender (TCP) channel name | AS400.MVS.TCP |
| I  | Receiver (SNA) channel name | MVS.AS400.SNA |
| J  | Receiver (TCP) channel name | MVS.AS400.TCP |

Connection to MQSeries for VSE/ESA

The values in this section of the table must match those used in Table 51 on page 554 as indicated.

| C  | Remote queue manager name | VSE |
| D  | Remote queue name         | VSE.REMOTEQ |
| E  | Queue name at remote system | VSE.LOCALQ |
| F  | Transmission queue name   | VSE |
| G  | Sender channel name       | AS400.VSE.SNA |
| H  | Receiver channel name     | VSE.AS400.SNA |

WebSphere MQ for iSeries sender-channel definitions using SNA

Local Queue
- Queue name: OS2
- Queue type: *LCL
- Usage: *TMQ

Remote Queue
- Queue name: OS2.REMOTEQ
- Queue type: *RMT
- Remote queue: OS2.LOCALQ
- Remote Queue Manager: OS2
- Transmission queue: OS2

Sender Channel
- Channel Name: AS400.OS2.SNA
- Channel Type: *SDR
- Transport type: *LU62
- Connection name: OS2CPIC
- Transmission queue: OS2
WebSphere MQ for iSeries receiver-channel definitions using SNA

Local Queue
Queue name : AS400.LOCALQ
Queue type : *LCL

Receiver Channel
Channel Name : OS2.AS400.SNA
Channel Type : *RCVR
Transport type : *LU62

WebSphere MQ for iSeries sender-channel definitions using TCP

Local Queue
Queue name : OS2
Queue type : *LCL
Usage : *TMQ

Remote Queue
Queue name : OS2.REMOTEQ
Queue type : *RMT
Remote queue : OS2.LOCALQ
Remote Queue Manager : OS2
Transmission queue : OS2

Sender Channel
Channel Name : AS400.OS2.TCP
Channel Type : *SDR
Transport type : *TCP
Connection name : os2.tcpip.hostname
Transmission queue : OS2

WebSphere MQ for iSeries receiver-channel definitions using TCP

Local Queue
Queue name : AS400.LOCALQ
Queue type : *LCL

Receiver Channel
Channel Name : OS2.AS400.TCP
Channel Type : *RCVR
Transport type : *TCP

Defining a queue
Type CRTMQMQ on the command line.
Fill in the two fields of this panel and press Enter. This causes another panel to appear, with entry fields for the other parameters you have. Defaults can be taken for all other queue attributes.

**Defining a channel**

Type CRTMQMCHL on the command line.
OS/400 configuration

Fill in the two fields of this panel and press Enter. Another panel is displayed on which you can specify the values for the other parameters given earlier. Defaults can be taken for all other channel attributes.
Chapter 43. Message channel planning example for iSeries

This chapter provides a detailed example of how to connect two iSeries queue managers together so that messages can be sent between them. The example illustrates the preparations needed to allow an application using queue manager QM1 to put messages on a queue at queue manager QM2. An application running on QM2 can retrieve these messages, and send responses to a reply queue on QM1.

The example illustrates the use of TCP/IP connections. The example assumes that channels are to be triggered to start when the first message arrives on the transmission queue they are servicing. You must start the channel initiator in order for triggering to work. To do this, use the STRMQMCHLI command.

This example uses SYSTEM.CHANNEL.INITQ as the initiation queue. This queue is already defined by WebSphere MQ. You can use a different initiation queue, but you will have to define it yourself and specify the name of the queue when you start the channel initiator.

What the example shows

The example uses the WebSphere MQ for iSeries command language.

![Diagram of message channel example](image)

Figure 118. The message channel example for WebSphere MQ for iSeries

It involves a payroll query application connected to queue manager QM1 that sends payroll query messages to a payroll processing application running on queue manager QM2. The payroll query application needs the replies to its queries sent back to QM1. The payroll query messages are sent from QM1 to QM2 on a sender-receiver channel called QM1.TO.QM2, and the reply messages are sent back from QM2 to QM1 on another sender-receiver channel called QM2.TO.QM1. Both of these channels are triggered to start as soon as they have a message to send to the other queue manager.

The payroll query application puts a query message to the remote queue “PAYROLL.QUERY” defined on QM1. This remote queue definition resolves to the local queue “PAYROLL” on QM2. In addition, the payroll query application specifies that the reply to the query is sent to the local queue “PAYROLL.REPLY” on QM1. The payroll processing application gets messages from the local queue “PAYROLL” on QM2, and sends the replies to wherever they are required; in this case, local queue “PAYROLL.REPLY” on QM1.

© Copyright IBM Corp. 1993, 2002
Both queue managers are assumed to be running on iSeries. In the example definitions, QM1 has a host address of 9.20.9.31 and is listening on port 1411, and QM2 has a host address of 9.20.9.32 and is listening on port 1412. The example assumes that these are already defined on your iSeries system, and are available for use.

The object definitions that need to be created on QM1 are:
- Remote queue definition, PAYROLL.QUERY
- Transmission queue definition, QM2 (default=remote queue manager name)
- Sender channel definition, QM1.TO.QM2
- Receiver channel definition, QM2.TO.QM1
- Reply-to queue definition, PAYROLL.REPLY

The object definitions that need to be created on QM2 are:
- Local queue definition, PAYROLL
- Transmission queue definition, QM1 (default=remote queue manager name)
- Sender channel definition, QM2.TO.QM1
- Receiver channel definition, QM1.TO.QM2

The connection details are supplied in the CONNAME attribute of the sender channel definitions.

You can see a diagram of the arrangement in Figure 118 on page 541.

**Queue manager QM1 example**

The following object definitions allow applications connected to queue manager QM1 to send request messages to a queue called PAYROLL on QM2, and to receive replies on a queue called PAYROLL.REPLY on QM1.

All the object definitions have been provided with the TEXT attributes. The other attributes supplied are the minimum required to make the example work. The attributes that are not supplied take the default values for queue manager QM1.

Run the following commands on queue manager QM1:

**Remote queue definition**

The CRTMQMQ command with the following attributes:

```
QNAME 'PAYROLL.QUERY'
QTYPE *RMT
TEXT 'Remote queue for QM2'
PUTENBL *YES
TMQNAME 'QM2' (default = remote queue manager name)
RMTQNAME 'PAYROLL'
RMTMQMNAME 'QM2'
```

**Note:** The remote queue definition is not a physical queue, but a means of directing messages to the transmission queue, QM2, so that they can be sent to queue manager QM2.
Transmission queue definition
The CRTMQMQ command with the following attributes:

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>QNAME</td>
<td>QM2</td>
</tr>
<tr>
<td>QTYPE</td>
<td>*LCL</td>
</tr>
<tr>
<td>TEXT</td>
<td>‘Transmission queue to QM2’</td>
</tr>
<tr>
<td>USAGE</td>
<td>*TMQ</td>
</tr>
<tr>
<td>PUTENBL</td>
<td>*YES</td>
</tr>
<tr>
<td>GETENBL</td>
<td>*YES</td>
</tr>
<tr>
<td>TRGENBL</td>
<td>*YES</td>
</tr>
<tr>
<td>TRGTYP</td>
<td>*FIRST</td>
</tr>
<tr>
<td>INITQNAME</td>
<td>SYSTEM.CHANNEL.INITQ</td>
</tr>
<tr>
<td>TRIGDATA</td>
<td>QM1.TO.QM2</td>
</tr>
</tbody>
</table>

When the first message is put on this transmission queue, a trigger message is sent to the initiation queue, SYSTEM.CHANNEL.INITQ. The channel initiator gets the message from the initiation queue and starts the channel identified in the named process.

Sender channel definition
The CRTMQMCHL command with the following attributes:

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>CHLNAME</td>
<td>QM1.TO.QM2</td>
</tr>
<tr>
<td>CHLTYP</td>
<td>*SDR</td>
</tr>
<tr>
<td>TRPTYP</td>
<td>*TCP</td>
</tr>
<tr>
<td>TEXT</td>
<td>‘Sender channel to QM2’</td>
</tr>
<tr>
<td>TMQNAME</td>
<td>QM2</td>
</tr>
<tr>
<td>CONNAME</td>
<td>‘9.20.9.32(1412)’</td>
</tr>
</tbody>
</table>
Planning example for iSeries

**Receiver channel definition**

The CRTMQMCHL command with the following attributes:

CHLNAME        QM2.TO.QM1  
CHLTYPE        *RCVR    
TRPTYPE        *TCP    
TEXT           ‘Receiver channel from QM2’

**Reply-to queue definition**

The CRTMQMQ command with the following attributes:

QNAME          PAYROLL.REPLY  
QTYPE          *LCL    
TEXT           ‘Reply queue for replies to query messages sent to QM2’
PUTENBL        *YES    
GETENBL        *YES

The reply-to queue is defined as PUT(ENABLED). This ensures that reply messages can be put to the queue. If the replies cannot be put to the reply-to queue, they are sent to the dead-letter queue on QM1 or, if this queue is not available, remain on transmission queue QM1 on queue manager QM2. The queue has been defined as GET(ENABLED) to allow the reply messages to be retrieved.

**Queue manager QM2 example**

The following object definitions allow applications connected to queue manager QM2 to retrieve request messages from a local queue called PAYROLL, and to put replies to these request messages to a queue called PAYROLL.REPLY on queue manager QM1.

You do not need to provide a remote queue definition to enable the replies to be returned to QM1. The message descriptor of the message retrieved from local queue PAYROLL contains both the reply-to queue and the reply-to queue manager names. Therefore, as long as QM2 can resolve the reply-to queue manager name to that of a transmission queue on queue manager QM2, the reply message can be sent. In this example, the reply-to queue manager name is QM1 and so queue manager QM2 simply requires a transmission queue of the same name.

All the object definitions have been provided with the TEXT attribute and are the minimum required to make the example work. The attributes that are not supplied take the default values for queue manager QM2.

Run the following commands on queue manager QM2:

**Local queue definition**

The CRTMQMQ command with the following attributes:

QNAME          PAYROLL  
QTYPE          *LCL    
TEXT           ‘Local queue for QM1 payroll details’
PUTENBL        *YES    
GETENBL        *YES

This queue is defined as PUT(ENABLED) and GET(ENABLED) for the same reason as the reply-to queue definition on queue manager QM1.
Transmission queue definition

The CRTMQMQ command with the following attributes:

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>QNAME</td>
<td>QM1</td>
</tr>
<tr>
<td>QTYPE</td>
<td>*LCL</td>
</tr>
<tr>
<td>TEXT</td>
<td>‘Transmission queue to QM1’</td>
</tr>
<tr>
<td>USAGE</td>
<td>*TMQ</td>
</tr>
<tr>
<td>PUTENBL</td>
<td>*YES</td>
</tr>
<tr>
<td>GETENBL</td>
<td>*YES</td>
</tr>
<tr>
<td>TRGENBL</td>
<td>*YES</td>
</tr>
<tr>
<td>TRGTYPE</td>
<td>*FIRST</td>
</tr>
<tr>
<td>INITQNAME</td>
<td>SYSTEM.CHANNEL.INITQ</td>
</tr>
<tr>
<td>TRGDATA</td>
<td>QM2.TO.QM1</td>
</tr>
</tbody>
</table>

When the first message is put on this transmission queue, a trigger message is sent to the initiation queue, SYSTEM.CHANNEL.INITQ. The channel initiator gets the message from the initiation queue and starts the channel identified in the trigger data.

Sender channel definition

The CRTMQMCHL command with the following attributes:

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>CHLNAME</td>
<td>QM2.TO.QM1</td>
</tr>
<tr>
<td>CHLTYPE</td>
<td>*SDR</td>
</tr>
<tr>
<td>TRPTYPE</td>
<td>*TCP</td>
</tr>
<tr>
<td>TEXT</td>
<td>‘Sender channel to QM1’</td>
</tr>
<tr>
<td>TMQNAME</td>
<td>QM1</td>
</tr>
<tr>
<td>CONNAME</td>
<td>‘9.20.9.31(1411)’</td>
</tr>
</tbody>
</table>

Receiver channel definition

The CRTMQMCHL command with the following attributes:

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>CHLNAME</td>
<td>QM1.TO.QM2</td>
</tr>
<tr>
<td>CHLTYPE</td>
<td>*RCVR</td>
</tr>
<tr>
<td>TRPTYPE</td>
<td>*TCP</td>
</tr>
<tr>
<td>TEXT</td>
<td>‘Receiver channel from QM1’</td>
</tr>
</tbody>
</table>

Running the example

When you have created the required objects, you must:
• Start the channel initiator for both queue managers
• Start the listener for both queue managers

The applications can then send messages to each other. The channels are triggered to start by the first message arriving on each transmission queue, so you do not need to issue the STRMQMCHL command.

For details about starting a channel initiator and a listener see Chapter 39, “Monitoring and controlling channels in WebSphere MQ for iSeries” on page 485.

Expanding this example

This example can be expanded by:
• Adding more queue and channel definitions to allow other applications to send messages between the two queue managers.
Planning example for iSeries

- Adding user exit programs on the channels to allow for link encryption, security checking, or additional message processing.
- Using queue manager aliases and reply-to queue aliases to understand more about how these can be used in the organization of your queue manager network.

For a version of this example that uses MQSC commands, see Chapter 27, “Message channel planning example for z/OS” on page 377.
Part 6. DQM in MQSeries for VSE/ESA

Chapter 44. Example configuration - MQSeries
for VSE/ESA ........................................... 549
Configuration parameters for an LU 6.2 connection 549
Configuration worksheet ................................ 549
Explanation of terms .................................. 551
Establishing an LU 6.2 connection .................... 552
  Defining a connection ................................ 552
  Defining a session ................................... 552
  Installing the new group definition ............... 553
  What next? ............................................ 553
Establishing a TCP connection ....................... 554
MQSeries for VSE/ESA configuration ............... 554
  Configuring channels ................................ 554
    MQSeries for VSE/ESA sender-channel
    definitions ....................................... 556
    MQSeries for VSE/ESA receiver-channel
    definitions ....................................... 557
  Defining a local queue ................................ 557
  Defining a remote queue ........................... 559
  Defining a SNA LU 6.2 sender channel ............ 561
  Defining a SNA LU6.2 receiver channel .......... 562
  Defining a TCP/IP sender channel ................ 564
  Defining a TCP receiver channel ................ 565
DQM in MQSeries for VSE/ESA
Chapter 44. Example configuration - MQSeries for VSE/ESA

This chapter gives an example of how to set up communication links from MQSeries for VSE/ESA to WebSphere MQ products on the following platforms:

- OS/2
- Windows
- AIX
- Compaq Tru64 UNIX
- HP-UX
- AT&T GIS UNIX\(^{10}\)
- Solaris
- OS/400
- z/OS, OS/390, or MVS/ESA without CICS

It describes the parameters needed for an LU 6.2 and TCP connection. Once the connection is established, you need to define some channels to complete the configuration. This is described in "MQSeries for VSE/ESA configuration" on page 554.

Configuration parameters for an LU 6.2 connection

Table 50 presents a worksheet listing all the parameters needed to set up communication from VSE/ESA to one of the other WebSphere MQ platforms. The worksheet shows examples of the parameters, which have been tested in a working environment, and leaves space for you to fill in your own values. An explanation of the parameter names follows the worksheet. Use the worksheet in this chapter in conjunction with the worksheet in the chapter for the platform to which you are connecting.

Configuration worksheet

Use the following worksheet to record the values you will use for this configuration. Where numbers appear in the Reference column they indicate that the value must match that in the appropriate worksheet elsewhere in this book. The examples that follow in this chapter refer back to the values in the ID column of this table. The entries in the Parameter Name column are explained in "Explanation of terms" on page 551.

Table 50. Configuration worksheet for VSE/ESA using APPC

<table>
<thead>
<tr>
<th>ID</th>
<th>Parameter Name</th>
<th>Reference</th>
<th>Example Used</th>
<th>User Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Network ID</td>
<td>NETID</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Node name</td>
<td>VSEPU</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Local LU name</td>
<td>VSELU</td>
<td></td>
<td>MQ01</td>
</tr>
<tr>
<td>4</td>
<td>Local Transaction Program name</td>
<td>MQ01</td>
<td>400074511092</td>
<td>MQ01</td>
</tr>
<tr>
<td>5</td>
<td>LAN destination address</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Connection to an OS/2 system

The values in this section of the table must match those used in the table for OS/2, as indicated.

---

10. This platform has become NCR UNIX SVR4 MP-RAS, R3.0

© Copyright IBM Corp. 1993, 2002
### Table 50. Configuration worksheet for VSE/ESA using APPC (continued)

<table>
<thead>
<tr>
<th>ID</th>
<th>Parameter Name</th>
<th>Reference</th>
<th>Example Used</th>
<th>User Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>Connection name</td>
<td></td>
<td>OS2</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>Group name</td>
<td></td>
<td>EXAMPLE</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>Session name</td>
<td></td>
<td>OS2SESS</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>Netname</td>
<td></td>
<td>OS2LU</td>
<td></td>
</tr>
</tbody>
</table>

**Connection to a Windows system**

The values in this section of the table must match those used in the table for Windows NT, as indicated.

<table>
<thead>
<tr>
<th>ID</th>
<th>Parameter Name</th>
<th>Reference</th>
<th>Example Used</th>
<th>User Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>Connection name</td>
<td></td>
<td>WNT</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>Group name</td>
<td></td>
<td>EXAMPLE</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>Session name</td>
<td></td>
<td>WNTSESS</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>Netname</td>
<td></td>
<td>WINNTLU</td>
<td></td>
</tr>
</tbody>
</table>

**Connection to an AIX system**

The values in this section of the table must match those used in the table for AIX, as indicated.

<table>
<thead>
<tr>
<th>ID</th>
<th>Parameter Name</th>
<th>Reference</th>
<th>Example Used</th>
<th>User Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>Connection name</td>
<td></td>
<td>AIX</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>Group name</td>
<td></td>
<td>EXAMPLE</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>Session name</td>
<td></td>
<td>AIXSESS</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>Netname</td>
<td></td>
<td>AIXLU</td>
<td></td>
</tr>
</tbody>
</table>

**Connection to an HP-UX system**

The values in this section of the table must match those used in the table for HP-UX, as indicated.

<table>
<thead>
<tr>
<th>ID</th>
<th>Parameter Name</th>
<th>Reference</th>
<th>Example Used</th>
<th>User Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>Connection name</td>
<td></td>
<td>HPUX</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>Group name</td>
<td></td>
<td>EXAMPLE</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>Session name</td>
<td></td>
<td>HPUXSESS</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>Netname</td>
<td></td>
<td>HPUXLU</td>
<td></td>
</tr>
</tbody>
</table>

**Connection to an AT&T GIS UNIX system**

The values in this section of the table must match those used in the table for GIS UNIX, as indicated.

<table>
<thead>
<tr>
<th>ID</th>
<th>Parameter Name</th>
<th>Reference</th>
<th>Example Used</th>
<th>User Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>Connection name</td>
<td></td>
<td>GIS</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>Group name</td>
<td></td>
<td>EXAMPLE</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>Session name</td>
<td></td>
<td>GISSESS</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>Netname</td>
<td></td>
<td>GISLU</td>
<td></td>
</tr>
</tbody>
</table>

**Connection to a Solaris system**

The values in this section of the table must match those used in the table for Solaris, as indicated.

<table>
<thead>
<tr>
<th>ID</th>
<th>Parameter Name</th>
<th>Reference</th>
<th>Example Used</th>
<th>User Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>Connection name</td>
<td></td>
<td>SOL</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>Group name</td>
<td></td>
<td>EXAMPLE</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>Session name</td>
<td></td>
<td>SOLSESS</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>Netname</td>
<td></td>
<td>SOLARLU</td>
<td></td>
</tr>
</tbody>
</table>

**Connection to an iSeries system**

The values in this section of the table must match those used in the table for iSeries, as indicated.

<table>
<thead>
<tr>
<th>ID</th>
<th>Parameter Name</th>
<th>Reference</th>
<th>Example Used</th>
<th>User Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>Connection name</td>
<td></td>
<td>AS4</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>Group name</td>
<td></td>
<td>EXAMPLE</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>Session name</td>
<td></td>
<td>AS4SESS</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>Netname</td>
<td></td>
<td>AS4000LU</td>
<td></td>
</tr>
</tbody>
</table>

**Connection to a z/OS system without CICS**

The values in this section of the table must match those used in the table for z/OS, as indicated.

<table>
<thead>
<tr>
<th>ID</th>
<th>Parameter Name</th>
<th>Reference</th>
<th>Example Used</th>
<th>User Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>Connection name</td>
<td></td>
<td>MVS</td>
<td></td>
</tr>
</tbody>
</table>
### Table 50. Configuration worksheet for VSE/ESA using APPC (continued)

<table>
<thead>
<tr>
<th>ID</th>
<th>Parameter Name</th>
<th>Reference</th>
<th>Example Used</th>
<th>User Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>Group name</td>
<td></td>
<td>EXAMPLE</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>Session name</td>
<td></td>
<td>MVSSESS</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>Netname</td>
<td></td>
<td>MVSLU</td>
<td></td>
</tr>
</tbody>
</table>

#### Explanation of terms

1. **Network ID**
   This is the unique ID of the network to which you are connected. Your system administrator will tell you this value.

2. **Node name**
   This is the name of the SSCP which owns the CICS/VSE region.

3. **Local LU name**
   This is the unique VTAM APPLID of this CICS/VSE region.

4. **Transaction Program name**
   WebSphere MQ applications trying to converse with this queue manager will specify a transaction name for the program to be run at the receiving end. This will have been defined on the channel definition at the sender. MQSeries for VSE/ESA uses a name of MQ01.

5. **LAN destination address**
   This is the LAN destination address that your partner nodes will use to communicate with this host. It is usually the address of the 3745 on the same LAN as the partner node.

6. **Connection name**
   This is a 4-character name by which each connection will be individually known in CICS RDO.

7. **Group name**
   You choose your own 8-character name for this value. Your system may already have a group defined for connections to partner nodes. Your system administrator will give you a value to use.

8. **Session name**
   This is an 8-character name by which each session will be individually known. For clarity we use the connection name, concatenated with 'SESS'.

9. **Netname**
   This is the LU name of the WebSphere MQ queue manager on the system with which you are setting up communication.
Establishing a connection

Establishing an LU 6.2 connection

This example is for a connection to an OS/2 system. The steps are the same whatever platform you are using; change the values as appropriate.

Defining a connection

1. At a CICS command line type CEDA DEF CONN(connection name) GROUP(group name) (where connection name is \textcolor{red}{6} and group name is \textcolor{red}{7}). For example:
   
   CEDA DEF CONN(OS2) GROUP(EXAMPLE)

2. Press Enter to define a connection to CICS.

   \begin{verbatim}
   DEF CONN(OS2) GROUP(EXAMPLE)
   OVERTYPE TO MODIFY
   CEDA DEFINE
   Connection  : OS2
   Group       : EXAMPLE
   DESCRIPTION =>
   CONNECTION IDENTIFIERS
   Netname     => OS2LU
   INDsys      =>
   REMOTE ATTRIBUTES
   REMOTESYSTEM =>
   REMOTENAME  =>
   CONNECTION PROPERTIES
   ACCESSMETHO = Vtam | IRC | INDIRECT | XM
   PROTOCOL     => Appc | LU61
   SINGLESESS  = No | Yes
   DATASTREAM  => User | 3270 | 3SC | STRFIELD | LMS
   RECORDFORMAT => U | Vb
   OPERATIONAL PROPERTIES
   + AUTOCONNECT => Yes | No | Yes | All
   I New group EXAMPLE created.
   
   DEFINE SUCCESSFUL
   TIME: 16.49.30 DATE: 96.054
   PF 1 HELP 2 COM 3 END 6 CRSR 7 SBH 8 SFH 9 MSG 10 SB 11 SF 12 CNCL
   \end{verbatim}

3. On the panel change the Netname field in the CONNECTION IDENTIFIERS section to be the LU name (\textcolor{red}{9}) of the target system.

4. In the CONNECTION PROPERTIES section set the ACCESSMETHOD field to VTAM and the PROTOCOL to Appc.

5. Press Enter to make the change.

Defining a session

1. At a CICS command line type CEDA DEF SESS(session name) GROUP(group name) (where session name is \textcolor{red}{8} and group name is \textcolor{red}{7}). For example:
   
   CEDA DEF SESS(OS2SESS) GROUP(EXAMPLE)

2. Press Enter to define a session for the connection.
3. In the SESSION IDENTIFIERS section of the panel specify the Connection name (6) in the Connection field and set the MOdename to #INTER.

4. In the SESSION PROPERTIES section set the Protocol to Appc and the MAXimum field to 008, 004.

Installing the new group definition

1. At a CICS command line type CEDA INS GROUP(group name) 7.

2. Press Enter to install the new group definition.

   Note: If this connection group is already in use you may get severe errors reported. If this happens, take the existing connections out of service, retry the above group installation, and then set the connections in service using the following commands:
   a. CEMT I CONN
   b. CEMT S CONN(*) OUTS
   c. CEDA INS GROUP(group name)
   d. CEMT S CONN(*) INS

What next?

The LU 6.2 connection is now established. You are ready to complete the configuration. Go to "MQSeries for VSE/ESA configuration" on page 554.
Establishing a TCP connection

TCP connections do not require the configuration of additional profiles as does the LU 6.2 protocol. Instead, MQSeries for VSE/ESA processes the listener program during startup.

The listener program waits for remote TCP connection requests. As these are received, the listener starts the receiver MCA to process the remote connection. When the remote connection is received from a client program, the receiver MCA starts the server program.

Note: There is one server process for each client connection.

Provided that the listener is active and TCP is active in a VSE partition, TCP connections can be established.

MQSeries for VSE/ESA configuration

Configuring MQSeries for VSE/ESA involves the following tasks:

- Configuring channels
- Defining a local queue
- Defining a remote queue
- Defining a sender channel
- Defining a receiver channel

Configuring channels

Examples are given for connecting MQSeries for VSE/ESA and MQSeries for OS/2 Warp. If you wish connect to another WebSphere MQ platform use the appropriate set of values from the table in place of those for OS/2.

Note: The words in bold are user-specified and reflect the names of WebSphere MQ objects used throughout these examples. If you change the names used here, ensure that you also change the other references made to these objects throughout this book. All others are keywords and should be entered as shown.

Refer to the sections “Defining a local queue” on page 557 and “Defining a remote queue” on page 559 for details of how to create queue definitions, and “Defining a SNA LU 6.2 sender channel” on page 561 and “Defining a SNA LU6.2 receiver channel” on page 562 for details of how to create channels.

### Table 51. Configuration worksheet for MQSeries for VSE/ESA

<table>
<thead>
<tr>
<th>ID</th>
<th>Parameter Name</th>
<th>Reference</th>
<th>Example Used</th>
<th>User Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>Queue Manager Name</td>
<td></td>
<td>VSE</td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>Local queue name</td>
<td></td>
<td>VSE.LOCALQ</td>
<td></td>
</tr>
</tbody>
</table>

**Definition for local node**

**Connection to MQSeries for OS/2 Warp**

The values in this section of the table must match those used in the worksheet table for OS/2, as indicated.

<p>| C  | Remote queue manager name      | A          | OS2          |            |
| D  | Remote queue name              |           | OS2.REMOTEQ  |            |
| E  | Queue name at remote system    | B          | OS2.LOCALQ   |            |
| F  | Transmission queue name        |           | OS2          |            |
| G  | Sender channel name            |           | VSE.OS2.SNA  |            |</p>
<table>
<thead>
<tr>
<th>ID</th>
<th>Parameter Name</th>
<th>Reference</th>
<th>Example Used</th>
<th>User Value</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Receiver channel name</td>
<td></td>
<td>OS2.VSE.SNA</td>
<td></td>
</tr>
</tbody>
</table>

**Connection to WebSphere MQ for Windows**

The values in this section of the table must match those used in the worksheet table for Windows NT, as indicated.

| C  | Remote queue manager name          | WINNT     |              |            |
|    | Remote queue name                  |           | WINNT.REMOTEQ|            |
| E  | Queue name at remote system        |           | WINNT.LOCALQ|            |
| F  | Transmission queue name            | WINNT     |              |            |
| G  | Sender channel name                | VSE.WINNT.SNA |            |            |
| I  | Receiver channel name              | WINNT.VSE.SNA |            |            |

**Connection to WebSphere MQ for AIX**

The values in this section of the table must match those used in the worksheet table for AIX, as indicated.

| C  | Remote queue manager name          | AIX       |              |            |
|    | Remote queue name                  |           | AIX.REMOTEQ  |            |
| E  | Queue name at remote system        |           | AIX.LOCALQ   |            |
| F  | Transmission queue name            | AIX       |              |            |
| G  | Sender channel name                | VSE.AIX.SNA |            |            |
| I  | Receiver channel name              | AIX.VSE.SNA |            |            |

**Connection to MQSeries for Compaq Tru64 UNIX**

The values in this section of the table must match those used in the worksheet table for UNIX, as indicated.

| C  | Remote queue manager name          | DECUX     |              |            |
|    | Remote queue name                  |           | DECUX.REMOTEQ|            |
| E  | Queue name at remote system        |           | DECUX.LOCALQ|            |
| F  | Transmission queue name            | DECUX     |              |            |
| H  | Sender (TCP) channel name          | DECUX.VSE.TCP |            |            |
| I  | Receiver channel name              | VSE.DECUX.TCP |            |            |

**Connection to WebSphere MQ for HP-UX**

The values in this section of the table must match those used in the worksheet table for HP-UX, as indicated.

| C  | Remote queue manager name          | HPUX      |              |            |
|    | Remote queue name                  |           | HPUX.REMOTEQ |            |
| E  | Queue name at remote system        |           | HPUX.LOCALQ  |            |
| F  | Transmission queue name            | HPUX      |              |            |
| G  | Sender channel name                | VSE.HPUX.SNA |            |            |
| I  | Receiver channel name              | HPUX.VSE.SNA |            |            |

**Connection to MQSeries for AT&T GIS UNIX**

The values in this section of the table must match those used in the worksheet table for GIS UNIX, as indicated.

| C  | Remote queue manager name          | GIS       |              |            |
|    | Remote queue name                  |           | GIS.REMOTEQ  |            |
| E  | Queue name at remote system        |           | GIS.LOCALQ   |            |
| F  | Transmission queue name            | GIS       |              |            |
| G  | Sender channel name                | VSE.GIS.SNA |            |            |
| I  | Receiver channel name              | GIS.VSE.SNA |            |            |

**Connection to WebSphere MQ for Solaris**

The values in this section of the table must match those used in the worksheet table for Solaris, as indicated.
### Table 51. Configuration worksheet for MQSeries for VSE/ESA (continued)

<table>
<thead>
<tr>
<th>ID</th>
<th>Parameter Name</th>
<th>Reference</th>
<th>Example Used</th>
<th>User Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>C</td>
<td>Remote queue manager name</td>
<td>SOLARIS</td>
<td>SOLARIS</td>
<td></td>
</tr>
<tr>
<td>D</td>
<td>Remote queue name</td>
<td>SOLARIS.REMOTEQ</td>
<td>SOLARIS.LOCALQ</td>
<td>SOLARIS</td>
</tr>
<tr>
<td>E</td>
<td>Queue name at remote system</td>
<td>SOLARIS.LOCALQ</td>
<td>VSE.SOLARIS.SNA</td>
<td>SOLARIS.VSE.SNA</td>
</tr>
<tr>
<td>F</td>
<td>Transmission queue name</td>
<td>SOLARIS</td>
<td></td>
<td></td>
</tr>
<tr>
<td>G</td>
<td>Sender channel name</td>
<td>VSE.SOLARIS.SNA</td>
<td>SOLARIS.VSE.SNA</td>
<td></td>
</tr>
<tr>
<td>I</td>
<td>Receiver channel name</td>
<td>VSE.SOLARIS.SNA</td>
<td>SOLARIS.VSE.SNA</td>
<td></td>
</tr>
</tbody>
</table>

**Connection to WebSphere MQ for iSeries**

The values in this section of the table must match those used in the worksheet table for iSeries, as indicated.

| C  | Remote queue manager name | AS400  |            |            |
| D  | Remote queue name | AS400.REMOTEQ | AS400.LOCALQ |            |
| E  | Queue name at remote system | AS400.LOCALQ | VSE.AS400.SNA | AS400.VSE.SNA |
| F  | Transmission queue name | AS400   |            |            |
| G  | Sender channel name | VSE.AS400.SNA |            |            |
| I  | Receiver channel name | AS400.VSE.SNA |            |            |

**Connection to WebSphere MQ for z/OS without CICS**

The values in this section of the table must match those used in the worksheet table for z/OS, as indicated.

| C  | Remote queue manager name | MVS  |            |            |
| D  | Remote queue name | MVS.REMOTEQ | MVS.LOCALQ |            |
| E  | Queue name at remote system | MVS.LOCALQ | VSE.MVS.SNA |            |
| F  | Transmission queue name | MVS   |            |            |
| G  | Sender channel name | VSE.MVS.SNA |            |            |
| I  | Receiver channel name | VSE.MVS.SNA |            |            |

For TCP, the sender channel name **G** and the receiver channel name **I**, in the preceding table, can be VSE.sys.tcp and sys.VSE.TCP respectively.

In both cases sys represents the remote system name, for example, OS2. Therefore, in this case, **G** becomes VSE.OS2.TCP and **I** becomes OS2.VSE.TCP.

### MQSeries for VSE/ESA sender-channel definitions

**Local Queue**

- Object Type : L
- Object Name : OS2
- Usage Mode: T (Transmission)

**Remote Queue**

- Object Type : R
- Object Name : OS2.REMOTEQ
- Remote QUEUE Name : OS2.LOCALQ
- Remote QM Name : OS2
- Transmission Name : OS2

**Sender Channel**

- Channel name : VSE.OS2.SNA
- Channel type : S (Sender)
- Transmission queue name : OS2
- Remote Task ID : MQTP
- Connection name : OS2
MQSeries for VSE/ESA receiver-channel definitions

Local Queue
Object type : QLOCAL
Object Name : VSE.LOCALQ
Usage Mode : N (Normal)

Receiver Channel
Channel name : OS2.VSE.SNA
Channel type : R (Receiver)

Defining a local queue
1. Run the MQSeries master terminal transaction MQMT.

2. Select option 1 to configure.
3. Select option 2 to work with queue definitions.

4. Select an Object type of L and specify the name of the queue.

5. Press PF5.

6. Press PF5 again.
7. Specify the name of a CICS file to store messages for this queue.

8. If you are creating a transmission queue, specify a **Usage Mode** of **T**, a **Program ID** of **MQPSEND**, and a **Channel Name**<\texttt{<G>}>.

   For a normal queue specify a **Usage Mode** of **N**.

9. Press PF5 again.

**Defining a remote queue**

1. Run the MQSeries master terminal transaction MQMT.

2. Select option 1 to configure.
3. Select option 2 to work with queue definitions.

4. Select an Object type of R and specify the name of the queue.
5. Press PF5.
6. Specify a remote queue name, remote queue manager name, and transmission queue name.

7. Press PF5.

**Defining a SNA LU 6.2 sender channel**

1. Run the MQSeries master terminal transaction MQMT.

2. Select option 1 to configure.
3. Select option 3 to work with channel definitions.

4. Complete the parameter fields as indicated, specifically the fields Name, Type, Partner, Transmission Queue Name, and TP Name. All other parameters can be entered as shown. Note that the default value for sequence number wrap is 999999, whereas for Version 2 products, this value defaults to 999999999.

5. Press PF5.

Defining a SNA LU6.2 receiver channel

1. Run the MQSeries master terminal transaction MQMT.
2. Select option 1 to configure.

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>Global System Definition</td>
</tr>
<tr>
<td>2.</td>
<td>Queue Definitions</td>
</tr>
<tr>
<td>3.</td>
<td>Channel Definitions</td>
</tr>
</tbody>
</table>

Option: 

Please enter one of the options listed.

3. Select option 3 to work with channel definitions.
4. Complete the parameter fields as indicated, specifically the field **Channel name**. All other parameters can be entered as shown.

5. Press PF5.

### Defining a TCP/IP sender channel

To define a TCP/IP sender channel, carry out the following procedure:

1. Run the MQSeries master terminal transaction MQMT.
2. Select option 1 to configure.
3. Select option 3 to work with channel definitions. The screen shown in [Figure 119 on page 565](#) is displayed. Follow the instructions after the figure to complete the required parameter fields:
4. Complete the parameter fields as follows:
   - Channel name – /SF580000G/SF590000 on the configuration worksheet.
   - Partner – should contain the IP address of the remote host, for example, 1.20.33.444.
   - Port – the port number must match the port number configured for the remote host. This is configured in the global system definition of the remote host. The default port number for MQSeries for VSE/ESA is 1414.
   - Transmission queue name – /SF580000F/SF590000 on the configuration worksheet.
   - Protocol – enter T for TCP.
   - Channel type – enter S for sender.

Notes:
   a. The TP Name is not used by TCP channels.
   b. Ensure that the parameter field values match the values of the receiver channel definition of the same name on the remote host.

5. Press PF5 (Add) to add the new channel definition.

**Defining a TCP receiver channel**

To define a TCP receiver channel, carry out the following procedure:
1. Run the MQSeries master terminal transaction MQMT.
2. Select option 1 to configure.
3. Select option 3 to work with channel definitions. The screen shown in [Figure 119](#) is displayed.
4. Complete the parameter fields as follows:
   - Channel name – / on the configuration worksheet.
   - Protocol – enter T for TCP.
   - Channel type – enter S for sender.

![Figure 119. Channel configuration panel](image-url)
VSE/ESA configuration

Notes:

a. The Partner and Port are not required for a TCP receiver channel.
b. The TP Name is not used by TCP channels.
c. Ensure that the parameter field values match the values of the sender channel definition of the same name on the remote host.

5. Press PF5 (Add) to add the new channel definition.
Part 7. Further intercommunication considerations

Chapter 45. Channel-exit programs

What are channel-exit programs? 571
Processing overview 572
Channel security exit programs 574
Channel send and receive exit programs 578
Channel send exit programs — reserving space 581
How you reserve space and use it 581
What happens at the receiving end of the channel 581
Multiple send exits 581
Channel message exit programs 581
Channel message retry exit program 583
Channel auto-definition exit program 583
Transport-retry exit program 584
Writing and compiling channel-exit programs 585
WebSphere MQ for z/OS without CICS 587
WebSphere MQ for z/OS using CICS 588
WebSphere MQ for iSeries 589
MQSeries for OS/2 Warp 589
Windows 3.1 client 591
WebSphere MQ for Windows server, WebSphere MQ client for Windows 591
WebSphere MQ for AIX 593
MQSeries for Compaq OpenVMS Alpha 595
MQSeries for Compaq Tru64 UNIX 596
WebSphere MQ for HP-UX 597
MQSeries for AT&T GIS UNIX 598
WebSphere MQ for Solaris 599
MQSeries for SINIX and DC/OSx 600
MQSeries for Compaq NonStop Kernel 600
Building and using channel exit functions 601
Supplied channel-exit programs using DCE security services 604
What do the DCE channel-exit programs do? 604
How do the DCE channel-exit programs work? 605
How to use the DCE channel-exit programs 607
Setup for DCE 607
The supplied exit code 608
Using DCE channel exits with the runmqlsr listener program 610

Chapter 46. Channel-exit calls and data structures

MQ_CHANNEL_EXIT – Channel exit 611
Syntax 612
Parameters 614
ChannelExitParms (MQCXP) – input/output 614
ChannelDefinition (MQCD) – input/output 614
DataLength (MQLONQ) – input/output 614
AgentBufferLength (MQLONQ) – input 615
AgentBuffer (MQBYTE×AgentBufferLength) – input/output 615
ExitBufferLength (MQLONQ) – input/output 616
ExitBufferAddr (MQPTR) – input/output 616
Usage notes 616
C invocation 616
COBOL invocation 617
PL/I invocation 617
RPG invocation (ILE) 617
RPG invocation (OPM) 618
System/390 assembler invocation 618
MQ_CHANNEL_AUTO_DEF_EXIT – Channel auto-definition exit 619
Syntax 619
Parameters 619
ChannelExitParms (MQCXP) – input/output 619
ChannelDefinition (MQCD) – input/output 619
Usage notes 619
C invocation 620
COBOL invocation 620
RPG invocation (ILE) 620
RPG invocation (OPM) 620
System/390 assembler invocation 620
MQ_TRANSPORT_EXIT – Transport retry exit 622
Syntax 622
Parameters 622
ExitParms (MQTXP) – input/output 622
DestAddressLength (MQLONQ) – input 622
DestAddress (MQCHAR×DestAddressLength) – input 622
Usage notes 622
C invocation 622
MQXWAIT – Wait in exit 624
Syntax 624
Parameters 624
Hconn (MQHCONN) – input 624
WaitDesc (MQXWD) – input/output 624
CompCode (MQLONQ) – output 624
Reason (MQLONQ) – output 624
C invocation 625
System/390 assembler invocation 625
MQCD – Channel definition 626
Exit name fields 628
Fields 628
ChannelName (MQCHAR20) 628
Version (MQLONQ) 629
ChannelType (MQLONQ) 630
TransportType (MQLONQ) 630
Desc (MQCHAR64) 631
QMgrName (MQCHAR48) 631
XmitQName (MQCHAR48) 631
ShortConnectionName (MQCHAR20) 631
MCAName (MQCHAR20) 632
ModeName (MQCHAR8) 632
TpName (MQCHAR64) 632
BatchSize (MQLONQ) 632
DiscInterval (MQLONQ) 632
ShortRetryCount (MQLONQ) 633
ShortRetryInterval (MQLONQ) 633
LongRetryCount (MQLONQ) 633

SSPI security exit 610

Data definition files 612
MQ_CHANNEL_EXIT – Channel exit 614
Syntax 614
Parameters 614
ChannelExitParms (MQCXP) – input/output 614
ChannelDefinition (MQCD) – input/output 614
DataLength (MQLONQ) – input/output 614
AgentBufferLength (MQLONQ) – input 615
AgentBuffer (MQBYTE×AgentBufferLength) – input/output 615
ExitBufferLength (MQLONQ) – input/output 616
ExitBufferAddr (MQPTR) – input/output 616
Usage notes 616
C invocation 616
COBOL invocation 617
PL/I invocation 617
RPG invocation (ILE). 617
RPG invocation (OPM) 618
System/390 assembler invocation 618

MQ_CHANNEL_AUTO_DEF_EXIT – Channel auto-definition exit 619
Syntax 619
Parameters 619
ChannelExitParms (MQCXP) – input/output 619
ChannelDefinition (MQCD) – input/output 619
Usage notes 619
C invocation 620
COBOL invocation 620
RPG invocation (ILE) 620
RPG invocation (OPM) 620
System/390 assembler invocation 620
MQ_TRANSPORT_EXIT – Transport retry exit 622
Syntax 622
Parameters 622
ExitParms (MQTXP) – input/output 622
DestAddressLength (MQLONQ) – input 622
DestAddress (MQCHAR×DestAddressLength) – input 622
Usage notes 622
C invocation 622
MQXWAIT – Wait in exit 624
Syntax 624
Parameters 624
Hconn (MQHCONN) – input 624
WaitDesc (MQXWD) – input/output 624
CompCode (MQLONQ) – output 624
Reason (MQLONQ) – output 624
C invocation 625
System/390 assembler invocation 625
MQCD – Channel definition 626
Exit name fields 628
Fields 628
ChannelName (MQCHAR20) 628
Version (MQLONQ) 629
ChannelType (MQLONQ) 630
TransportType (MQLONQ) 630
Desc (MQCHAR64) 631
QMgrName (MQCHAR48) 631
XmitQName (MQCHAR48) 631
ShortConnectionName (MQCHAR20) 631
MCAName (MQCHAR20) 632
ModeName (MQCHAR8) 632
TpName (MQCHAR64) 632
BatchSize (MQLONQ) 632
DiscInterval (MQLONQ) 632
ShortRetryCount (MQLONQ) 633
ShortRetryInterval (MQLONQ) 633
LongRetryCount (MQLONQ) 633

© Copyright IBM Corp. 1993, 2002
### Further intercommunication considerations

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>LongRetryInterval (MQLONG)</td>
<td></td>
<td>633</td>
</tr>
<tr>
<td>SecurityExit (MQCHARn)</td>
<td></td>
<td>634</td>
</tr>
<tr>
<td>MsgExit (MQCHARn)</td>
<td></td>
<td>634</td>
</tr>
<tr>
<td>SendExit (MQCHARn)</td>
<td></td>
<td>634</td>
</tr>
<tr>
<td>ReceiveExit (MQCHARn)</td>
<td></td>
<td>635</td>
</tr>
<tr>
<td>SeqNumberWrap (MQLONG)</td>
<td></td>
<td>635</td>
</tr>
<tr>
<td>MaxMsgLength (MQLONG)</td>
<td></td>
<td>635</td>
</tr>
<tr>
<td>PutAuthority (MQLONG)</td>
<td></td>
<td>636</td>
</tr>
<tr>
<td>DataConversion (MQLONG)</td>
<td></td>
<td>635</td>
</tr>
<tr>
<td>SecurityUserData (MQCHAR32)</td>
<td></td>
<td>636</td>
</tr>
<tr>
<td>SendUserData (MQCHAR32)</td>
<td></td>
<td>636</td>
</tr>
<tr>
<td>ReceiveUserData (MQCHAR32)</td>
<td></td>
<td>637</td>
</tr>
<tr>
<td>UserIdentifier (MQCHAR12)</td>
<td></td>
<td>637</td>
</tr>
<tr>
<td>Password (MQCHAR12)</td>
<td></td>
<td>637</td>
</tr>
<tr>
<td>MCAUserIdentifier (MQCHAR12)</td>
<td></td>
<td>637</td>
</tr>
<tr>
<td>MCAType (MQLONG)</td>
<td></td>
<td>638</td>
</tr>
<tr>
<td>ConnectionName (MQCHAR264)</td>
<td></td>
<td>639</td>
</tr>
<tr>
<td>RemoteUserIdentifier (MQCHAR12)</td>
<td></td>
<td>639</td>
</tr>
<tr>
<td>RemotePassword (MQCHAR12)</td>
<td></td>
<td>640</td>
</tr>
<tr>
<td>MsgRetryExit (MQCHARn)</td>
<td></td>
<td>640</td>
</tr>
<tr>
<td>MsgRetryUserData (MQCHAR32)</td>
<td></td>
<td>641</td>
</tr>
<tr>
<td>MsgRetryCount (MQLONG)</td>
<td></td>
<td>641</td>
</tr>
<tr>
<td>HeartbeatInterval (MQLONG)</td>
<td></td>
<td>642</td>
</tr>
<tr>
<td>BatchInterval (MQLONG)</td>
<td></td>
<td>642</td>
</tr>
<tr>
<td>NonPersistentMsgSpeed (MQLONG)</td>
<td></td>
<td>643</td>
</tr>
<tr>
<td>StrucLength (MQLONG)</td>
<td></td>
<td>644</td>
</tr>
<tr>
<td>ExitNameLength (MQLONG)</td>
<td></td>
<td>644</td>
</tr>
<tr>
<td>ExitDataLength (MQLONG)</td>
<td></td>
<td>644</td>
</tr>
<tr>
<td>MsgExitsDefined (MQLONG)</td>
<td></td>
<td>645</td>
</tr>
<tr>
<td>SendExitsDefined (MQLONG)</td>
<td></td>
<td>645</td>
</tr>
<tr>
<td>ReceiveExitsDefined (MQLONG)</td>
<td></td>
<td>645</td>
</tr>
<tr>
<td>MsgBoxPtr (MQPTR)</td>
<td></td>
<td>645</td>
</tr>
<tr>
<td>MMsgPtr (MQPTR)</td>
<td></td>
<td>645</td>
</tr>
<tr>
<td>SendPtr (MQPTR)</td>
<td></td>
<td>645</td>
</tr>
<tr>
<td>ReceivePtr (MQPTR)</td>
<td></td>
<td>647</td>
</tr>
<tr>
<td>ClusterPtr (MQPTR)</td>
<td></td>
<td>648</td>
</tr>
<tr>
<td>ClustersDefined (MQLONG)</td>
<td></td>
<td>648</td>
</tr>
<tr>
<td>NetworkPriority (MQLONG)</td>
<td></td>
<td>648</td>
</tr>
<tr>
<td>LongMCAUserIdLength (MQLONG)</td>
<td></td>
<td>648</td>
</tr>
<tr>
<td>LongRemoteUserldLength (MQLONG)</td>
<td></td>
<td>648</td>
</tr>
<tr>
<td>LongRemoteUserIdPtr (MQPTR)</td>
<td></td>
<td>649</td>
</tr>
<tr>
<td>LongRemoteUserldPtr (MQPTR)</td>
<td></td>
<td>649</td>
</tr>
<tr>
<td>MCAUserSecurity (MQBYTE40)</td>
<td></td>
<td>649</td>
</tr>
<tr>
<td>RemoteSecurity (MQBYTE40)</td>
<td></td>
<td>650</td>
</tr>
<tr>
<td>SSLCipherSpec (MQCHAR32)</td>
<td></td>
<td>650</td>
</tr>
<tr>
<td>SSLPeerNamePtr (MQPTR)</td>
<td></td>
<td>650</td>
</tr>
<tr>
<td>SSLPeerNameLength (MQLONG)</td>
<td></td>
<td>650</td>
</tr>
<tr>
<td>SSLClientAuth (MQLONG)</td>
<td></td>
<td>651</td>
</tr>
<tr>
<td>KeepAliveInterval (MQLONG)</td>
<td></td>
<td>651</td>
</tr>
<tr>
<td>LocalAddress (MQCHAR48)</td>
<td></td>
<td>651</td>
</tr>
<tr>
<td>BatchHeartbeat (MQLONG)</td>
<td></td>
<td>652</td>
</tr>
</tbody>
</table>

### MQTXP – Transport exit parameter

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Strucld (MQCHAR4)</td>
<td></td>
<td>664</td>
</tr>
<tr>
<td>Version (MQLONG)</td>
<td></td>
<td>667</td>
</tr>
<tr>
<td>ExitId (MQLONG)</td>
<td></td>
<td>667</td>
</tr>
<tr>
<td>ExitReason (MQLONG)</td>
<td></td>
<td>668</td>
</tr>
<tr>
<td>ExitResponse (MQLONG)</td>
<td></td>
<td>670</td>
</tr>
<tr>
<td>ExitResponse2 (MQLONG)</td>
<td></td>
<td>671</td>
</tr>
<tr>
<td>Feedback (MQLONG)</td>
<td></td>
<td>673</td>
</tr>
<tr>
<td>MaxSegmentLength (MQLONG)</td>
<td></td>
<td>673</td>
</tr>
<tr>
<td>MaxUserArea (MQBYTE16)</td>
<td></td>
<td>673</td>
</tr>
<tr>
<td>DataLength (MQLONG)</td>
<td></td>
<td>675</td>
</tr>
<tr>
<td>ExitNumber (MQLONG)</td>
<td></td>
<td>675</td>
</tr>
<tr>
<td>ExitSpace (MQLONG)</td>
<td></td>
<td>676</td>
</tr>
<tr>
<td>SessionId (MQLONG)</td>
<td></td>
<td>683</td>
</tr>
<tr>
<td>GroupId (MQLONG)</td>
<td></td>
<td>683</td>
</tr>
<tr>
<td>DataId (MQLONG)</td>
<td></td>
<td>683</td>
</tr>
<tr>
<td>ExitResponse (MQLONG)</td>
<td></td>
<td>683</td>
</tr>
<tr>
<td>Feedback (MQLONG)</td>
<td></td>
<td>684</td>
</tr>
</tbody>
</table>

### MQXWD – Exit wait descriptor

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Strucld (MQCHAR4)</td>
<td></td>
<td>665</td>
</tr>
<tr>
<td>Version (MQLONG)</td>
<td></td>
<td>665</td>
</tr>
<tr>
<td>ExitId (MQLONG)</td>
<td></td>
<td>665</td>
</tr>
<tr>
<td>ExitReason (MQLONG)</td>
<td></td>
<td>668</td>
</tr>
<tr>
<td>ExitResponse (MQLONG)</td>
<td></td>
<td>670</td>
</tr>
<tr>
<td>ExitResponse2 (MQLONG)</td>
<td></td>
<td>671</td>
</tr>
<tr>
<td>Feedback (MQLONG)</td>
<td></td>
<td>673</td>
</tr>
</tbody>
</table>

### Chapter 47. Problem determination in DQM

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Error message from channel control</td>
<td>687</td>
</tr>
<tr>
<td>Ping</td>
<td>687</td>
</tr>
<tr>
<td>Dead-letter queue considerations</td>
<td>688</td>
</tr>
<tr>
<td>Validation checks</td>
<td>688</td>
</tr>
<tr>
<td>In-doubt relationship</td>
<td>689</td>
</tr>
</tbody>
</table>
Further intercommunication considerations

- Channel startup negotiation errors
- When a channel refuses to run
  - Triggered channels
  - Conversion failure
- Network problems
  - Adopting an MCA
  - Registration time for DDNS
  - Dial-up problems
- Retrying the link
  - Retry considerations
  - Shared channel recovery on z/OS
- Data structures
- User exit problems
- Disaster recovery
- Channel switching
- Connection switching
- Client problems
  - Terminating clients
- Error logs
  - Error logs for OS/2, Windows and Windows 98 client
  - Error logs on UNIX systems
  - Error logs on DOS, Windows 3.1, and Windows 95 clients
  - Error logs on z/OS
  - Error logs on MQSeries for VSE/ESA
  - Error logs on MQSeries for Compaq NonStop Kernel

Part 7. Further intercommunication considerations
Further intercommunication considerations
Chapter 45. Channel-exit programs

This chapter discusses WebSphere MQ channel-exit programs. This is product-sensitive programming interface information. The following topics are covered:

- “What are channel-exit programs?”
- “Writing and compiling channel-exit programs” on page 585
- “Supplied channel-exit programs using DCE security services” on page 604

Message channel agents (MCAs) can also call data-conversion exits; these are discussed in the WebSphere MQ Application Programming Guide.

Note: Channel exit programs are not supported on DOS or VSE/ESA.

What are channel-exit programs?

Channel-exit programs are called at defined places in the processing carried out by MCA programs.

Some of these user-exit programs work in complementary pairs. For example, if a user-exit program is called by the sending MCA to encrypt the messages for transmission, the complementary process must be functioning at the receiving end to reverse the process.

The different types of channel-exit program are described below. Table 52 shows the types of channel exit that are available for each channel type.

Table 52. Channel exits available for each channel type

<table>
<thead>
<tr>
<th>Channel Type</th>
<th>Message exit</th>
<th>Message-retry exit</th>
<th>Receive exit</th>
<th>Security exit</th>
<th>Send exit</th>
<th>Auto-definition exit</th>
<th>Transport-retry exit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sender channel</td>
<td>✓</td>
<td></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>Server channel</td>
<td>✓</td>
<td></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>Cluster-sender channel</td>
<td>✓</td>
<td></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Receiver channel</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Requester channel</td>
<td>✓</td>
<td></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Cluster-receiver channel</td>
<td>✓</td>
<td></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>Client-connection channel</td>
<td>✓</td>
<td></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>Server-connection channel</td>
<td>✓</td>
<td></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
</tr>
</tbody>
</table>

Notes™:
1. The message-retry exit does not apply to WebSphere MQ for z/OS.
2. The auto-definition exit applies to WebSphere MQ for AIX, HP-UX, iSeries, Solaris, Windows, and z/OS (cluster-sender channels only), and MQSeries for Compaq Tru64 UNIX, and OS/2 Warp V5.1 and WebSphere MQ for.
3. The transport-retry exit applies to WebSphere MQ for AIX.
Channel-exit programs

If you are going to run channel exits on a client, you cannot use the MQSERVER environment variable. Instead, create and reference a client channel definition table as described in the WebSphere MQ Clients book.

Processing overview

On startup, the MCAs exchange a startup dialog to synchronize processing. Then they switch to a data exchange that includes the security exits; these must end successfully for the startup phase to complete and to allow messages to be transferred.

The security check phase is a loop, as shown in Figure 120.

![Diagram of Security exit loop](image)

Figure 120. Security exit loop

During the message transfer phase, the sending MCA gets messages from a transmission queue, calls the message exit, calls the send exit, and then sends the message to the receiving MCA, as shown in Figure 121 on page 573.
The receiving MCA receives a message from the communications link, calls the receive exit, calls the message exit, and then puts the message on the local queue, as shown in Figure 122. (The receive exit can be called more than once before the message exit is called.)
Channel security exit programs

You can use security exit programs to verify that the partner at the other end of a channel is genuine.

Channel security exit programs are called at the following places in an MCA’s processing cycle:

- At MCA initiation and termination.
- Immediately after the initial data negotiation is finished on channel startup. The receiver or server end of the channel may initiate a security message exchange with the remote end by providing a message to be delivered to the security exit at the remote end. It may also decline to do so. The exit program is re-invoked to process any security message received from the remote end.
- Immediately after the initial data negotiation is finished on channel startup. The sender or requester end of the channel processes a security message received from the remote end, or initiates a security exchange when the remote end cannot. The exit program is re-invoked to process all subsequent security messages that may be received.

A requester channel never gets called with MQXCC_INIT_SEC. The channel notifies the server that it has a security exit program, and the server then has the opportunity to initiate a security exit. If it does not have one, it sends a null security flow to allow the requester to call its exit program.

Note: You are recommended to avoid sending zero-length security messages.

WebSphere MQ for AIX, HP-UX, Linux, Solaris, and Windows, and MQSeries V5.1 for Compaq Tru64 UNIX, and OS/2 Warp, and the WebSphere MQ client for Windows 95 and Windows 98 supply a security exit program that uses the DCE security services. See “Supplied channel-exit programs using DCE security services” on page 604.

Examples of the data exchanged by security-exit programs are illustrated in figures 123 through 126. These examples show the sequence of events that occur involving the receiver’s security exit (left-hand column) and the sender’s security exit (right-hand column). Successive rows in the figures represent the passage of time. In some cases, the events at the receiver and sender are not correlated, and therefore can occur at the same time or at different times. In other cases, an event at one exit program results in a complementary event occurring later at the other exit program. For example, in Figure 123 on page 575:

1. The receiver and sender are each invoked with MQXR_INIT, but these invocations are not correlated and can therefore occur at the same time or at different times.

2. The receiver is next invoked with MQXR_INIT_SEC, but returns MQXCC_OK which requires no complementary event at the sender exit.

3. The sender is next invoked with MQXR_INIT_SEC. This is not correlated with the invocation of the receiver with MQXR_INIT_SEC. The sender returns MQXCC_SEND_SEC_MSG, which causes a complementary event at the receiver exit.

4. The receiver is subsequently invoked with MQXR_SEC_MSG, and returns MQXCC_SEND_SEC_MSG, which causes a complementary event at the sender exit.

5. The sender is subsequently invoked with MQXR_SEC_MSG, and returns MQXCC_OK which requires no complementary event at the receiver exit.
<table>
<thead>
<tr>
<th>Receiver exit</th>
<th>Sender exit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Invoked with MQXR_INIT</td>
<td>Invoked with MQXR_INIT</td>
</tr>
<tr>
<td>Responds with MQXCC_OK</td>
<td>Responds with MQXCC_OK</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>Invoked with MQXR_INIT_SEC</td>
<td>Invoked with MQXR_INIT_SEC</td>
</tr>
<tr>
<td>Responds with MQXCC_OK</td>
<td>Responds with MQXCC_OK</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>Invoked with MQXR_SEC_MSG</td>
<td>Invoked with MQXR_SEC_MSG</td>
</tr>
<tr>
<td>Responds with MQXCC_SEND_SEC_MSG</td>
<td>Responds with MQXCC_SEND_SEC_MSG</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Invoked with MQXR_SEC_MSG</td>
</tr>
<tr>
<td></td>
<td>Responds with MQXCC_OK</td>
</tr>
</tbody>
</table>

*Message transfer begins*

*Figure 123. Sender-initiated exchange with agreement*
## Channel-exit programs

<table>
<thead>
<tr>
<th>Receiver exit</th>
<th>Sender exit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Invoked with MQXR_INIT</td>
<td>Invoked with MQXR_INIT</td>
</tr>
<tr>
<td>Responds with MQXCC_OK</td>
<td>Responds with MQXCC_OK</td>
</tr>
<tr>
<td></td>
<td>Invoked with MQXR_INIT_SEC</td>
</tr>
<tr>
<td></td>
<td>Responds with MQXCC_OK</td>
</tr>
<tr>
<td>Invoked with MQXR_SEC</td>
<td></td>
</tr>
<tr>
<td>Responds with MQXCC_OK</td>
<td>Invoked with MQXR_SEC_MSG</td>
</tr>
<tr>
<td></td>
<td>Responds with MQXCC_SUPPRESS_FUNCTION</td>
</tr>
<tr>
<td>Invoked with MQXR_SEC_MSG</td>
<td></td>
</tr>
<tr>
<td>Responds with MQXCC_OK</td>
<td></td>
</tr>
<tr>
<td>Channel closes</td>
<td></td>
</tr>
<tr>
<td>Invoked with MQXR_TERM</td>
<td>Invoked with MQXR_TERM</td>
</tr>
<tr>
<td>Responds with MQXCC_OK</td>
<td>Responds with MQXCC_OK</td>
</tr>
</tbody>
</table>

*Figure 124. Sender-initiated exchange with no agreement*
### Channel-exit programs

#### Figure 125. Receiver-initiated exchange with agreement

<table>
<thead>
<tr>
<th>Receiver exit</th>
<th>Sender exit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Invoked with MQXR_INIT</td>
<td>Invoked with MQXR_INIT</td>
</tr>
<tr>
<td>Responds with MQXCC_OK</td>
<td>Responds with MQXCC_OK</td>
</tr>
<tr>
<td>Invoked with MQXR_INIT_SEC</td>
<td>Invoked with MQXR_SEC_MSG</td>
</tr>
<tr>
<td>Responds with MQXCC_SEND_SEC_MSG</td>
<td>Responds with MQXCC_SEND_SEC_MSG</td>
</tr>
<tr>
<td>Invoked with MQXR_SEC_MSG</td>
<td>Invoked with MQXR_TERM</td>
</tr>
<tr>
<td>Responds with MQXCC_OK</td>
<td>Responds with MQXCC_OK</td>
</tr>
</tbody>
</table>

*Message transfer begins*

<table>
<thead>
<tr>
<th>Receiver exit</th>
<th>Sender exit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Invoked with MQXR_TERM</td>
<td>Invoked with MQXR_TERM</td>
</tr>
<tr>
<td>Responds with MQXCC_OK</td>
<td>Responds with MQXCC_OK</td>
</tr>
</tbody>
</table>

#### Figure 126. Receiver-initiated exchange with no agreement

<table>
<thead>
<tr>
<th>Receiver exit</th>
<th>Sender exit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Invoked with MQXR_INIT</td>
<td>Invoked with MQXR_INIT</td>
</tr>
<tr>
<td>Responds with MQXCC_OK</td>
<td>Responds with MQXCC_OK</td>
</tr>
<tr>
<td>Invoked with MQXR_INIT_SEC</td>
<td>Invoked with MQXR_SEC_MSG</td>
</tr>
<tr>
<td>Responds with MQXCC_SEND_SEC_MSG</td>
<td>Responds with MQXCC_OK</td>
</tr>
<tr>
<td>Invoked with MQXR_SEC_MSG</td>
<td>Responds with MQXCC_OK</td>
</tr>
<tr>
<td>Responds with MQXCC_SUPPRESS_FUNCTION</td>
<td><strong>Channel closes</strong></td>
</tr>
</tbody>
</table>

---

Chapter 45. Channel-exit programs 577
The channel security exit program is passed an agent buffer containing the security data, excluding any transmission headers, generated by the security exit. This may be any suitable data so that either end of the channel is able to perform security validation.

The security exit program at both the sending and receiving end of the message channel may return one of four response codes to any call:

- Security exchange ended with no errors
- Suppress the channel and close down
- Send a security message to the corresponding security exit at the remote end
- Send a security message and demand a reply (this does not apply on z/OS when using CICS)

Notes:
1. The channel security exits usually work in pairs. When you define the appropriate channels, make sure that compatible exit programs are named for both ends of the channel.
2. In OS/400, security exit programs that have been compiled with “Use adopted authority” (USEADPAUT=YES) have the ability to adopt QMQM or QMQMADM authority. Take care that the exit does not exploit this feature to pose a security risk to your system.

Channel send and receive exit programs

You can use the send and receive exits to perform tasks such as data compression and decompression. WebSphere MQ for AIX, iSeries, HP-UX, Linux, Solaris, z/OS without CICS, and Windows, and MQSeries V5.1 for Compaq Tru64 UNIX, and OS/2 Warp, and with WebSphere MQ clients, you can specify a list of send and receive exit programs to be run in succession.

Channel send and receive exit programs are called at the following places in an MCA’s processing cycle:

- The send and receive exit programs are called for initialization at MCA initiation and for termination at MCA termination.
- The send exit program is invoked at either end of the channel, immediately before a transmission is sent over the link.
- The receive exit program is invoked at either end of the channel, immediately after a transmission has been taken from the link.

Note: For WebSphere MQ for z/OS using CICS, only the security exit is called at MCA initiation; other exits are called with the ExitReason parameter set to MQXR-INIT when the first message is sent across the channel.

There may be many transmissions for one message transfer, and there could be many iterations of the send and receive exit programs before a message reaches the message exit at the receiving end.

The channel send and receive exit programs are passed an agent buffer containing the transmission data as sent or received from the communications link. For send exit programs, the first eight bytes of the buffer are reserved for use by the MCA, and must not be changed. If the program returns a different buffer, then these first eight bytes must exist in the new buffer. The format of data presented to the exit programs is not defined.

A good response code must be returned by send and receive exit programs. Any other response will cause an MCA abnormal end (abend).
Note: Do not issue an MQGET, MQPUT, or MQPUT1 call within syncpoint from a send or receive exit.

WebSphere MQ for AIX, HP-UX, Linux, Solaris, and Windows, and MQSeries V5.1 for Compaq Tru64 UNIX, and OS/2 Warp, and the WebSphere MQ client for Windows 95 and Windows 98 supply send and receive exit programs that use the DCE encryption security services. See “Supplied channel-exit programs using DCE security services” on page 604.

Notes:
1. Send and receive exits usually work in pairs. For example a send exit may compress the data and a receive exit decompress it, or a send exit may encrypt the data and a receive exit decrypt it. When you define the appropriate channels, make sure that compatible exit programs are named for both ends of the channel.

2. Channel send and receive exits may be called for message segments other than for application data, for example, status messages. They are not called during the startup dialog, nor the security check phase.

3. Although message channels send messages in one direction only, channel-control data flows in both directions, and these exits are available in both directions, also. However, some of the initial channel startup data flows are exempt from processing by any of the exits.

4. There are circumstances in which send and receive exits could be invoked out of sequence; for example, if you are running a series of exit programs or if you are also running security exits. Then, when the receive exit is first called upon to process data, it may receive data that has not passed through the corresponding send exit. If the receive exit were just to perform the operation, for example decompression, without first checking that it was really required, the results would be unexpected.

You should code your send and receive exits in such a way that the receive exit can check that the data it is receiving has been processed by the corresponding send exit. The recommended way to do this is to code your exit programs so that:
- The send exit sets the value of the ninth byte of data to 0 and shifts all the data along one byte, before performing the operation. (The first eight bytes are reserved for use by the MCA.)
- If the receive exit receives data that has a 0 in byte 9, it knows that the data has come from the send exit. It removes the 0, performs the complementary operation, and shifts the resulting data back by one byte.
- If the receive exit receives data that has something other than 0 in byte 9, it assumes that the send exit has not run, and sends the data back to the caller unchanged.

When using security exits, if the channel is ended by the security exit it is possible that a send exit may be called without the corresponding receive exit. One way to prevent this from being a problem is to code the security exit to set a flag, in MQCD.SecurityUserData or MQCD.SendUserData, for example, when the exit decides to end the channel. Then the send exit should check this field, and process the data only if the flag is not set. This prevents the send exit from unnecessarily altering the data, and thus prevents any conversion errors that could occur if the security exit received altered data.
5. In the case of MQI channels for clients, byte 10 of the agent buffer identifies the API call in use when the send or receive exit is called. This is useful for identifying which channel flows include user data and may require processing such as encryption or digital signing. Table 53 shows the data that appears in byte 10 of the channel flow when an API call is being processed.

**Note:** These are not the only values of this byte. There are other reserved values.

Table 53. Identifying API calls

<table>
<thead>
<tr>
<th>API call</th>
<th>Value of byte 10</th>
</tr>
</thead>
<tbody>
<tr>
<td>MQCONN request</td>
<td>X'81'</td>
</tr>
<tr>
<td>MQCONN reply</td>
<td>X'91'</td>
</tr>
<tr>
<td>MQDISC request</td>
<td>X'82'</td>
</tr>
<tr>
<td>MQDISC reply</td>
<td>X'92'</td>
</tr>
<tr>
<td>MQOPEN request</td>
<td>X'83'</td>
</tr>
<tr>
<td>MQOPEN reply</td>
<td>X'93'</td>
</tr>
<tr>
<td>MQCLOSE request</td>
<td>X'84'</td>
</tr>
<tr>
<td>MQCLOSE reply</td>
<td>X'94'</td>
</tr>
<tr>
<td>MQGET request</td>
<td>X'85'</td>
</tr>
<tr>
<td>MQGET reply</td>
<td>X'95'</td>
</tr>
<tr>
<td>MQPUT request</td>
<td>X'86'</td>
</tr>
<tr>
<td>MQPUT reply</td>
<td>X'96'</td>
</tr>
<tr>
<td>MQPUT1 request</td>
<td>X'87'</td>
</tr>
<tr>
<td>MQPUT1 reply</td>
<td>X'97'</td>
</tr>
<tr>
<td>MQSET request</td>
<td>X'88'</td>
</tr>
<tr>
<td>MQSET reply</td>
<td>X'98'</td>
</tr>
<tr>
<td>MQINQ request</td>
<td>X'89'</td>
</tr>
<tr>
<td>MQINQ reply</td>
<td>X'99'</td>
</tr>
<tr>
<td>MQCMIT request</td>
<td>X'8A'</td>
</tr>
<tr>
<td>MQCMIT reply</td>
<td>X'9A'</td>
</tr>
<tr>
<td>MQBACK request</td>
<td>X'8B'</td>
</tr>
<tr>
<td>MQBACK reply</td>
<td>X'9B'</td>
</tr>
</tbody>
</table>

**Notes:**

a. The connection between the client and server is initiated by the client application using MQCONN. Therefore, for this command in particular, there will be several other network flows. This also applies to MQDISC that terminates the network connection.

b. MQCONNX is treated in the same way as MQCONN for the purposes of the client-server connection.

c. If a large distribution list is opened, there may be more than one network flow per MQOPEN call in order to pass all of the required data to the SVRCONN MCA.

d. If the message data exceeds the transmission segment size, there may be a large number of network flows per single API call.
Channel send exit programs — reserving space

You can use send and receive exits to transform the data before transmission. Channel send exit programs can add their own data about the transformation by reserving space in the transmission buffer. This data is processed by the receive exit program and then removed from the buffer. For example, you might want to encrypt the data and add a security key for decryption.

How you reserve space and use it

Channel exit programs are passed an MQCXP parameter block. A new field, ExitSpace, is added to the MQCXP structure, as described in “MQCXP – Channel exit parameter” on page 666. This parameter is not supported on z/OS.

When the send exit program is called for initialization, set the ExitSpace field to the number of bytes to be reserved. ExitSpace can be set only during initialization, that is when ExitReason has the value MQXR_INIT. When the send exit is invoked immediately before transmission, with ExitReason set to MQXR_XMIT, ExitSpace bytes are reserved in the transmission buffer.

The send exit need not use all of the reserved space. It can use less than ExitSpace bytes or, if the transmission buffer is not full, the exit can use more than the amount reserved. When setting the value of ExitSpace, you must leave at least 1 KB for message data in the transmission buffer. Note that channel performance can be affected if reserved space is used for large amounts of data.

What happens at the receiving end of the channel

Channel receive exit programs must be set up to be compatible with the corresponding send exits. Receive exits must know the number of bytes in the reserved space and must remove the data in that space.

Multiple send exits

You can specify a list of send and receive exit programs to be run in succession. WebSphere MQ maintains a total for the space reserved by all of the send exits. This total space must leave at least 1 KB for message data in the transmission buffer.

The following example shows how space is allocated for three send exits, called in succession:

1. When called for initialization:
   - Send exit A reserves 1 KB.
   - Send exit B reserves 2 KB.
   - Send exit C reserves 3 KB.
2. The maximum transmission size is 32 KB and the user data is 5 KB long.
3. Exit A is called with 5 KB of data; up to 27 KB are available, because 5 KB is reserved for exits B and C. Exit A adds 1 KB, the amount it reserved.
4. Exit B is called with 6 KB of data; up to 29 KB are available, because 3 KB is reserved for exit C. Exit B adds 1 KB, less than the 2 KB it reserved.
5. Exit C is called with 7 KB of data; up to 32 KB are available. Exit C adds 10 KB, more than the 3 KB it reserved. This is valid, because the total amount of data, 17 KB, is less than the 32 KB maximum.

Channel message exit programs

You can use the channel message exit for the following:
- Encryption on the link
- Validation of incoming user IDs
- Substitution of user IDs according to local policy
Channel-exit programs

- Message data conversion
- Journaling
- Reference message handling

WebSphere MQ for AIX, iSeries, HP-UX, Linux, Solaris, z/OS without CICS, and Windows, and MQSeries V5.1 for Compaq Tru64 UNIX, and OS/2 Warp, and with WebSphere MQ clients, you can specify a list of message exit programs to be run in succession.

Channel message exit programs are called at the following places in an MCA’s processing cycle:
- At MCA initiation and termination
- Immediately after a sending MCA has issued an MQGET call
- Before a receiving MCA issues an MQPUT call

The message exit is passed an agent buffer containing the transmission queue header, MQXQH, and the application message text as retrieved from the queue. (The format of MQXQH is given in the [WebSphere MQ Application Programming Reference](#) book.) If you use reference messages, that is messages that contain only a header which points to some other object that is to be sent, the message exit recognizes the header, MQRMH. It identifies the object, retrieves it in whatever way is appropriate appendes it to the header, and passes it to the MCA for transmission to the receiving MCA. At the receiving MCA, another message exit recognizes that this is a reference message, extracts the object, and passes the header on to the destination queue. See the [WebSphere MQ Application Programming Guide](#) for more information about reference messages and some sample message exits that handle them.

Message exits can return the following responses:
- Send the message (GET exit). The message may have been changed by the exit. (This returns MQXCC_OK.)
- Put the message on the queue (PUT exit). The message may have been changed by the exit. (This returns MQXCC_OK.)
- Do not process the message. The message is placed on the dead-letter queue (undelivered message queue) by the MCA.
- Close the channel.
- Bad return code, which causes the MCA to abend.

**Notes:**
1. Message exits are called just once for every complete message transferred, even when the message is split into parts.
2. In UNIX systems, if you provide a message exit for any reason the automatic conversion of user IDs to lowercase characters does not operate. See “User IDs on UNIX systems, Compaq OpenVMS Alpha” on page 125.
3. An exit runs in the same thread as the MCA itself. It also runs inside the same unit of work (UOW) as the MCA because it uses the same connection handle. Therefore, any calls made under syncpoint are committed or backed out by the channel at the end of the batch. For example, one channel message exit program can send notification messages to another and these messages will only be committed to the queue when the batch containing the original message is committed. Therefore, it is possible to issue syncpoint MQI calls from a channel message exit program.
Channel-exit programs

WebSphere MQ for AIX, HP-UX, Linux, Solaris, and Windows, and MQSeries V5.1 for Compaq Tru64 UNIX, and OS/2 Warp supplies a message exit program that uses the DCE security services. See "Supplied channel-exit programs using DCE security services" on page 604.

Channel message retry exit program

The channel message-retry exit is called when an attempt to open the target queue is unsuccessful. You can use the exit to determine under which circumstances to retry, how many times to retry, and how frequently. (This exit is not available on WebSphere MQ for z/OS.)

This exit is also called at the receiving end of the channel at MCA initiation and termination.

The channel message-retry exit is passed an agent buffer containing the transmission queue header, MQXQH, and the application message text as retrieved from the queue. The format of MQXQH is given in the WebSphere MQ Application Programming Reference book.

The exit is invoked for all reason codes; the exit determines for which reason codes it wants the MCA to retry, for how many times, and at what intervals. (The value of the message-retry count set when the channel was defined is passed to the exit in the MQCD, but the exit can ignore this.)

The MsgRetryCount field in MQCXP is incremented by the MCA each time the exit is invoked, and the exit returns either MQXCC_OK with the wait time contained in the MsgRetryInterval field of MQCXP, or MQXCC_SUPPRESS_FUNCTION. Retries continue indefinitely until the exit returns MQXCC_SUPPRESS_FUNCTION in the ExitResponse field of MQCXP. See "MQCXP – Channel exit parameter" on page 666 for information about the action taken by the MCA for these completion codes.

If all the retries are unsuccessful, the message is written to the dead-letter queue. If there is no dead-letter queue available, the channel stops.

If you do not define a message-retry exit for a channel and a failure occurs that is likely to be temporary, for example MQRC_Q_FULL, the MCA uses the message-retry count and message-retry intervals set when the channel was defined. If the failure is of a more permanent nature and you have not defined an exit program to handle it, the message is written to the dead-letter queue.

Channel auto-definition exit program

The channel auto-definition exit can be called when a request is received to start a receiver or server-connection channel but no channel definition exists. The exit applies to WebSphere MQ for AIX, HP-UX, iSeries, Solaris, and Windows, and MQSeries for Compaq Tru64 UNIX, and OS/2 Warp V5.1. You can use it to modify the supplied default definition for an automatically defined receiver or server-connection channel, SYSTEM.AUTO.RECEIVER or SYSTEM.AUTO.SVRCON. See "Auto-definition of receiver and server-connection channels" on page 60 for a description of how channel definitions can be created automatically.

The channel auto-definition exit can also be called when a request is received to start a cluster-sender channel. It can be called for cluster-sender and cluster-receiver channels to allow definition modification for this instance of the
Channel-exit programs

channel. In this case, the exit applies to WebSphere MQ for z/OS as well as WebSphere MQ for AIX, HP-UX, iSeries, Solaris, and Windows, and MQSeries for Compaq Tru64 UNIX, and OS/2 Warp V5.1. For more information about this, see the WebSphere MQ Queue Manager Clusters book.

As with other channel exits, the parameter list is:
MQ_CHANNEL_AUTO_DEF_EXIT (ChannelExitParms, ChannelDefinition)

ChannelExitParms are described in “MQCXXP – Channel exit parameter” on page 666. ChannelDefinition is described in “MQCD – Channel definition” on page 626.

MQCD contains the values that are used in the default channel definition if they are not altered by the exit. The exit may modify only a subset of the fields; see “MQ_CHANNEL_AUTO_DEF_EXIT – Channel auto-definition exit” on page 619. However, attempting to change other fields does not cause an error.

The channel auto-definition exit returns a response of either MQXCC_OK or MQXCC_SUPPRESS_FUNCTION. If neither of these is returned, the MCA continues processing as though MQXCC_SUPPRESS_FUNCTION were returned. That is, the auto-definition is abandoned, no new channel definition is created and the channel cannot start.

Transport-retry exit program

The transport-retry exit applies to WebSphere MQ for AIX, V5.3 and is supported on UDP only. It allows you to write a C-language retry exit. The exit allows your application to suspend data being sent on a channel when communication is not possible (for example, when a mobile user is traveling through a tunnel or is temporarily out of range of a transmitter).

The transport-retry exit can be associated with a monitor program that can assess whether the IP connection is available for sending data. The exit has to be built into a library that is included in the path in which you are operating.

The exit is normally called before a datagram is about to be sent but is also called to provide other useful signals.

The retry exit is called under five different conditions:
• When the WebSphere MQ channel is first initialized; the ExitReason variable is set to a value of MQXR_INIT.
• When the WebSphere MQ channel is shut down; the ExitReason variable is set to a value of MQXR_TERM.
• Before each datagram is sent; the ExitReason variable is set to a value of MQXR_RETRY.
• When the end of a batch of messages occurs; the ExitReason variable is set to a value of MQXR_END_BATCH.
• When an information datagram is received from the remote end of the link; the ExitReason variable is set to a value of MQXR_ACK_RECEIVED.

If you want to postpone sending a datagram in response to an ExitReason of MQXR_RETRY, you need to block returning from the exit until it is safe to send the datagram. In all other cases, the return from the exit should be immediate.

There are three possible return codes that can be set when returning from the exit:
Channel-exit programs

- MQXCC_OK — this is the normal response.
- MQXCC_CLOSE_CHANNEL — in response to an ExitReason of MQXR_RETRY, this will cause the channel to be closed.
- MQXCC_REQUEST_ACK — in response to an ExitReason of MQXR_RETRY, this will cause the datagram about to be sent to be modified so that it requests the remote end of the link to send an information datagram back to indicate that the node can be reached. If this datagram arrives the exit will be invoked again with an ExitReason of MQXR_ACK_RECEIVED.

Note: If the datagram fails to arrive at the remote node, for any reason, you must repeat the request on the next datagram that is sent.

The transport-retry exit name can be defined by the user, who can also change the name of the library that contains the exit. You configure the retry exit by editing the qm.ini file. For more information about editing these files, see the WebSphere MQ System Administration Guide book.

Writing and compiling channel-exit programs

Channel exits must be named in the channel definition. You can do this when you first define the channels, or you can add the information later using, for example, the MQSC command ALTER CHANNEL. You can also give the channel exit names in the MQCD channel data structure. The format of the exit name depends on your WebSphere MQ platform; see “MQCD – Channel definition” on page 626 or the WebSphere MQ Script (MQSC) Command Reference book for information.

If the channel definition does not contain a user-exit program name, the user exit is not called.

The channel auto-definition exit is the property of the queue manager, not the individual channel. In order for this exit to be called, it must be named in the queue manager definition. To alter a queue manager definition, use the MQSC command ALTER QMGR.

User exits and channel-exit programs are able to make use of all MQI calls, except as noted in the sections that follow. To get the connection handle, an MQCONN call must be issued, even though a warning, MQRC_ALREADY_CONNECTED, is returned because the channel itself is connected to the queue manager.

For exits on client-connection channels, the queue manager to which the exit tries to connect, depends on how the exit was linked. If the exit was linked with MQM.LIB (or QMQM/LIBMQM on iSeries) and you do not specify a queue manager name on the MQCONN call, the exit will try to connect to the default queue manager on your system. If the exit was linked with MQM.LIB (or QMQM/LIBMQM on iSeries) and you specify the name of the queue manager that was passed to the exit through the QMgrName field of MQCD, the exit tries to connect to that queue manager. If the exit was linked with MQIC.LIB or any other library, the MQCONN call will fail whether you specify a queue manager name or not.

Note: You are recommended to avoid issuing the following MQI calls in channel-exit programs:
- MQCMIT
- MQBACK
- MQBEGIN
Channel-exit programs

- MQDISC

An exit runs in the same thread as the MCA itself and uses the same connection handle. So, it runs inside the same UOW as the MCA and any calls made under syncpoint are committed or backed out by the channel at the end of the batch.

Therefore, a channel message exit could send notification messages that will only be committed to that queue when the batch containing the original message is committed. So, it is possible to issue syncpoint MQI calls from a channel message exit.

Channel-exit programs should not modify the Channel data structure (MQCD). They can actually change the BatchSize parameter and a security exit can set the MCAUserIdentifier parameter, but ChannelType and ChannelName must not be changed.

Also, for programs written in C, non-reentrant C library function should not be used in a channel-exit program.

All exits are called with a channel exit parameter structure (MQCXP), a channel definition structure (MQCD), a prepared data buffer, data length parameter, and buffer length parameter. The buffer length must not be exceeded:

- For message exits, you should allow for the largest message required to be sent across the channel, plus the length of the MQXQH structure.
- For send and receive exits, the largest buffer you should allow for is as follows:
  - LU 6.2:
    - OS/2 64 KB
    - Others 32 KB
  - TCP:
    - iSeries 16 KB
    - Others 32 KB
  - UDP:
    - 32 KB
  - NetBIOS:
    - DOS 4 KB
    - Others 64 KB
  - SPX:
    - 64 KB

  **Note:** The maximum usable length may be 2 bytes less than this. Check the value returned in MaxSegmentLength for details. For more information on MaxSegmentLength, see “MaxSegmentLength (MQLONG)” on page 673.

- For security exits, the distributed queuing facility allocates a buffer of 4000 bytes.
- On z/OS using CICS, all exits use the maximum transmission length for the channel, defined in the channel definition.
It is permissible for the exit to return an alternate buffer, together with the relevant parameters. See Chapter 45, “Channel-exit programs” on page 571 for call details.

**Note:** Before using a channel-exit program for the first time on WebSphere MQ for AIX, iSeries, HP-UX, Solaris, and Windows, and MQSeries V5.1 for Compaq Tru64 UNIX, and OS/2 Warp, you should relink it with threaded libraries to make it thread-safe.

WebSphere MQ for z/OS without CICS

The exits are invoked as if by a z/OS LINK, in:
- Non-authorized problem program state
- Primary address space control mode
- Non-cross-memory mode
- Non-access register mode
- 31-bit addressing mode

The link-edited modules must be placed in the data set specified by the CSQXLIB DD statement of the channel initiator address space procedure; the names of the load modules are specified as the exit names in the channel definition.

When writing channel exits for z/OS without CICS, the following rules apply:
- Exits must be written in assembler or C; if C is used, it must conform to the C systems programming environment for system exits, described in the z/OS C/C++ Programming Guide.
- Exits are loaded from the non-authorized libraries defined by a CSQXLIB DD statement. Providing CSQXLIB has DISP=SHR, exits can be updated while the channel initiator is running, with the new version used when the channel is restarted.
- Exits must be reentrant, and capable of running anywhere in virtual storage.
- Exits must reset the environment, on return, to that at entry.
- Exits must free any storage obtained, or ensure that it will be freed by a subsequent exit invocation.
  - For storage that is to persist between invocations, use the z/OS STORAGE service; there is no suitable service in C.
  - All MQI calls except MQCMIT/CSQBCMT and MQBACK/CSQBBAK are allowed. They must be contained after MQCONN (with a blank queue manager name). If these calls are used, the exit must be link-edited with the stub CSQXSTUB.
  - The exception to this rule is that security channel exits may issue commit and backout MQI calls. To do this, code the verbs CSQXCMT and CSQXBAK in place of MQCMIT/CSQBCMT and MQBACK/CSQBBAK.
Channel-exit programs

- Exits should not use any system services that could cause a wait, because this would severely impact the handling of some or all of the other channels. Many channels are run under a single TCB typically, if you do something in an exit that causes a wait and you do not use MQXWAIT, it will cause all these channels to wait. This will not give any functional problems, but might have an adverse effect on performance. Most SVCs involve waits, so you should avoid them, except for the following:
  - GETMAIN/FREEMAIN/STORAGE
  - LOAD/DELETE

In general, therefore, SVCs, PCs, and I/O should be avoided. Instead, the MQXWAIT call should be used.

- Exits should not issue ESTAEs or SPIEs, apart from in any subtasks they attach. This is because their error handling might interfere with the error handling performed by WebSphere MQ. This means that WebSphere MQ might not be able to recover from an error, or that your exit program might not receive all the error information.

- The MQXWAIT call (see "MQXWAIT – Wait in exit" on page 624) provides a wait service that allows waiting for I/O and other events; if this service is used, exits must not use the linkage stack.

For I/O and other facilities that do not provide non-blocking facilities or an ECB to wait on, a separate subtask should be ATTACHed, and its completion waited for by MQXWAIT; because of the overhead that this technique incurs, it is recommended that this be used only by the security exit.

- The MQDISC MQI call will not cause an implicit commit to occur within the exit program. A commit of the channel process is performed only when the channel protocol dictates.

The following exit samples are provided with WebSphere MQ for z/OS:

CSQ4BAX0
This sample is written in assembler, and illustrates the use of MQXWAIT.

CSQ4BCX1 and CSQ4BCX2
These samples are written in C and illustrate how to access the parameters.

WebSphere MQ for z/OS using CICS

In WebSphere MQ for z/OS using CICS, an exit program must be written in Assembler, C, COBOL, or PL/I. In CICS, the exits are invoked with EXEC CICS LINK with the parameters passed by pointers (addresses) in the CICS communication area (COMMAREA). The exit programs, named in the channel definitions, reside in a library in the DFHRPL concatenation. They must be defined in the CICS system definition file CSD, and must be enabled.

User-exit programs can also make use of CICS API calls, but you should not issue syncpoints because the results could influence units of work declared by the MCA.

Do not update any resources controlled by a resource manager other than WebSphere MQ for z/OS, including those controlled by CICS Transaction Server for z/OS.

Any non-WebSphere MQ for z/OS resources updated by an exit are committed, or backed out, at the next syncpoint issued by the channel program. If a sender is unable to synchronize with its partner, these CICS Transaction Server for z/OS resources are backed out even though WebSphere MQ for z/OS resources are held in-doubt until the next opportunity to re-synchronize.
WebSphere MQ for iSeries

The exit is a program object written in the C/400®, ILE COBOL/400® or ILE RPG/400® language. The exit program names and their libraries are named in the channel definition.

Observe the following conditions when creating and compiling an exit program:

- The program must be made thread safe and created with the C/400, ILE RPG/400, or ILE COBOL/400 compiler. For ILE RPG you must specify the THREAD(*SERIALIZE) control specification, and for ILE COBOL you must specify SERIALIZE for the THREAD option of the PROCESS statement. The programs must also be bound to the threaded WebSphere MQ libraries: QMQM/LIBMQM_R in the case of C/400 and ILE RPG/400, and AMQ0STUB_R in the case of ILE COBOL/400. For additional information about making RPG or COBOL applications thread safe, refer to the appropriate Programmer’s Guide for the language.

- WebSphere MQ for iSeries requires that the exit programs are enabled for teraspace support. (Teraspace is a form of shared memory introduced in OS/400 V4R4.) In the case of the ILE RPG and COBOL compilers, any programs compiled on OS/400 V4R4 or later are so enabled. In the case of C, the programs must be compiled with the TERASPACE(*YES *TSIFC) options specified on CRTCMOD or CRTBNDC commands.

- An exit returning a pointer to its own buffer space must ensure that the object pointed to exists beyond the timespan of the channel-exit program. In other words, the pointer cannot be the address of a variable on the program stack, nor of a variable in the program heap. Instead, the pointer must be obtained from the system. An example of this is a user space created in the user exit. To ensure that any data area allocated by the channel-exit program is still available for the MCA when the program ends, the channel exit must run in the caller’s activation group or a named activation group. Do this by setting the ACTGRP parameter on CRTPGM to a user-defined value or *CALLER. If the program is created in this way, the channel-exit program can allocate dynamic memory and pass a pointer to this memory back to the MCA.

MQSeries for OS/2 Warp

The exit is a DLL that must be written in C. To ensure that it can be loaded when required, specify the full path name in the DEFINE CHANNEL command, or if you are using Version 5.1 or later, enter the path name in the ExitPath stanza of the QM.INI file. The value in the ExitPath stanza of the QM.INI file defaults to c:\mqm\exits. You can change this value in QM.INI or you can override it by specifying a full path name on the DEFINE CHANNEL command.

Define a dummy MQStart() routine in the exit and specify MQStart as the entry point in the shared library. Figure 127 on page 590 shows how to set up entry to your program:
Channel-exit programs

```c
#include <cmqc.h>
#include <cmqxc.h>

void MQStart() {} /* dummy entry point - for consistency only */
void MQENTRY ChannelExit ( PMQVOID pChannelExitParms,
  PMQVOID pChannelDefinition,
  PMQLONG pDataLength,
  PMQLONG pAgentBufferLength,
  PMQVOID pAgentBuffer,
  PMQLONG pExitBufferLength,
  PMQPTR pExitBufferAddr)
{
  ... Insert code here
}
```

Figure 127. Sample source code for a channel exit on OS/2

Figure 128 shows a sample definition file that gives the entry point to the exit program.

```c
LIBRARY csqos2it INITINSTANCE TERMINSTANCE

PROTMODE

DESCRIPTION 'channel exit'

CODE SHARED LOADONCALL
DATA NONSHARED MULTIPLE

HEAPSIZE 4096
STACKSIZE 8192

EXPORTS
csqos2it;
```

Figure 128. Sample DEF file for a channel exit on OS/2

Use a make file like the one shown in Figure 129 on page 591 to compile and link your program to create the DLL.
Windows 3.1 client

The exit is a DLL that must be written in C. It must be placed in a directory pointed to by LIBPATH to ensure it can be loaded when required. Define a dummy MQStart() routine in the exit and specify MQStart as the entry point in the shared library. Figure 130 shows how to set up an entry to your program:

```c
#include <cmqc.h>
#include <cmqxc.h>

void MQStart() {} /* dummy entry point - for consistency only */
void MQENTRY ChannelExit ( PMQVOID pChannelExitParms,
   PMQVOID pChannelDefinition,
   PMQLONG pDataLength,
   PMQLONG pAgentBufferLength,
   PMQVOID pAgentBuffer,
   PMQLONG pExitBufferLength,
   PMQPTR pExitBufferAddr)
{
   ... Insert code here
}
```

Figure 130. Sample source code for a channel exit on Windows 3.1

WebSphere MQ for Windows server, WebSphere MQ client for Windows

The exit is a DLL that must be written in C.
Channel-exit programs

- On a WebSphere MQ for Windows server, specify the path name of the directory that holds the exit program on the Exits page of the queue manager properties (accessed from the WebSphere MQ Services snap-in).

- If the exit is on a Windows client, specify the path name on the All Queue Managers page of the WebSphere MQ properties (accessed from the WebSphere MQ services snap-in).

  The WebSphere MQ Services snap-in is described in the WebSphere MQ System Administration Guide manual.

- On a WebSphere MQ client for Windows 95, specify the path name in the ExitPath stanza of the MQS.INI file. You can change this value or you can override it by specifying a full path name on the DEFINE CHANNEL command.

- On a WebSphere MQ client for Windows 95, specify the path name in the ExitPath stanza of the MQS.INI file. You can change this value or you can override it by specifying a full path name on the DEFINE CHANNEL command.

Define a dummy MQStart() routine in the exit and specify MQStart as the entry point in the library. Figure 131 shows how to set up an entry to your program:

```c
#include <cmqc.h>
#include <cmqxc.h>

void MQStart() {}
void MQENTRY ChannelExit ( PMQCXP pChannelExitParms, PMQCD pChannelDefinition, PMQLONG pDataLength, PMQLONG pAgentBufferLength, PMQVOID pAgentBuffer, PMQLONG pExitBufferLength, PMQPTR pExitBufferAddr)
{
    ... Insert code here
}
```

**Figure 131. Sample source code for a channel exit on Windows**

In order to access the fields pointed to by pChannelExitParms and pChannelDefinition, you need to insert the following lines in your exit program:

```c
/* Variable definitions */
...
PMQCXP pParms;
PMQCD pChDef;
...  
/* Code */
...
pParms = (PMQCXP)pChannelExitParms;
pChDef = (PMQCD)pChannelDefinition;
```

The pointers pParms and pChDef can then be dereferenced to access individual fields.

When writing channel exits for these products using Visual C++, you should do the following:
- Add MQMVX.LIB to project as a source file.11

---

11. MQMVX.LIB is used for data conversion and is not available on client products.
Channel-exit programs

- Change the box labelled “Use Run-Time Library” from “Multithreaded” to “Multithreaded using DLL” in the project settings under C/C++ code generation.

- Do not change the box labelled “Entry-Point Symbol”. This box can be found in the project settings, under the Link tab, when you select Category and then Output.

- Write your own .DEF file; an example of this is shown in Figure 132.

```
LIBRARY exit

PROTMODE

DESCRIPTION 'Provides Retry and Channel exits'

CODE SHARED LOADONCALL
DATA NONSHARED MULTIPLE

HEAPSIZE  4096
STACKSIZE  8192

EXPORTS Retry
```

Figure 132. Sample DEF file for Windows NT, Windows 95, Windows 98, or Windows

WebSphere MQ for AIX

**Note:** Before you use an existing user exit for the first time on WebSphere MQ for AIX, you must recompile it to enable it to take advantage of thread-safe system calls. If your user exits use thread-unsafe system calls, you will need to modify them before using them on this platform.

The exit is a dynamically loaded object that must be written in C. To ensure that it can be loaded when required, specify the full path name in the DEFINE CHANNEL command or enter the path name in the ExitPath stanza of the QM.INI file. If the exit is on an AIX client, specify the path name in the ClientExitPath stanza of the MQS.INI file. The value in the ExitPath stanza of the QM.INI file or the ClientExitPath stanza of the MQS.INI file defaults to /var/mqm/exits. You can change this value or you can override it by specifying a full path name on the DEFINE CHANNEL command.

Define a dummy MQStart() routine in the exit and specify MQStart as the entry point in the module. Figure 133 on page 594 shows how to set up an entry to your program:
Channel-exit programs

```c
#include <cmqc.h>
#include <cmqxc.h>

void MQStart() {} /* dummy entry point - for consistency only */
void MQENTRY ChannelExit ( PMQVOID pChannelExitParms,
  PMQVOID pChannelDefinition,
  PMQLONG pDataLength,
  PMQLONG pAgentBufferLength,
  PMQVOID pAgentBuffer,
  PMQLONG pExitBufferLength,
  PMQPTR pExitBufferAddr)
{
  /* Insert code here */
}
```

Figure 133. Sample source code for a channel exit on AIX

Figure 134 shows the compiler and loader commands for channel-exit programs on AIX.

$ cc -c exit.c
$ ld -o exit exit.o -bE:exit.exp -H512 -T512 -e MQStart -bM:SRE
$ cp exit /usr/xmp/lib # (or wherever you require)

Figure 134. Sample compiler and loader commands for channel exits on AIX
Figure 136 shows a sample make file that can be used to build a WebSphere MQ exit program, and Figure 135 shows a sample export file for this make file.

**Note:** All functions that will be called by WebSphere MQ must be exported.

```bash
#!/csqaixit
MQStart
```

*Figure 135. Sample export file for AIX*

```bash
# MAKE FILE TO BUILD A WEBSPHERE MQ EXIT ON AIX
MQIDIR  = /usr/mqm
MQILIBDIR = $(MQIDIR)/lib
MQINCDIR = $(MQIDIR)/inc
LIBEXIT  = -lqm
CFLAGS  = -g -bloadmap:muck
ALL : CSQAIXIT
csqaixit: csqaixit.o
   xlc -L $(MQILIBDIR) $(LIBEXIT) csqaixit.o -o csqaixit \ 
   -bE:csqaixit.exp -H512 -T512 -e MQStart -bM:SRE
csqaixit.o : csqaixit.c
   xlc -c csqaixit.c \ 
   -I $(MQINCDIR)
```

*Figure 136. Sample make file for AIX*

**MQSeries for Compaq OpenVMS Alpha**

The user exit is a dynamically loaded image that can be shared, with its name taken from the format of the message. It must be written in C. The object’s name must be in uppercase, for example MYFORMAT. The shareable image must be placed in sys$share or a location defined by a logical name at executive level for it to be loaded.

User exits must be installed as known images. Figure 137 on page 596 shows how to set up an entry to your program:
Channel-exit programs

```c
#include <cmqc.h>
#include <cmqxc.h>

void MQStart(); /* dummy entry point */
void MQENTRY ChannelExit ( PMQVOID pChannelExitParms,
PMQVOID pChannelDefinition,
PMQLONG pDataLength,
PMQLONG pAgentBufferLength,
PMQVOID pAgentBuffer,
PMQLONG pExitBufferLength,
PMQPTR pExitBufferAddr)
{
... Insert code here
}
```

Figure 137. Sample source code for a channel exit on Compaq OpenVMS Alpha

In the example, MQSTART is the initialization routine entry point for the MYFORMAT shareable image. The names of the routines that are called by the exit must be made universal.

```
$ CC /INCLUDE_DIRECTORY=MQS_INCLUDE exitname.C
$ LINK /SHARE=SYS$SHARE:MYFORMAT exitname.OBJ,MYFORMAT/OPTIONS
```

The contents of MYFORMAT.OPT vary depending on what platform you are working on:

On AXP:
```
SYS$SHARE:MQM/SHAREABLE
SYMBOL_VECTOR=(MQSTART=PROCEDURE)
```

On VAX:
```
SYS$SHARE:MQM/SHAREABLE
UNIVERSAL=MQSTART
```

If you are using threaded applications linked with the pthread library, you must also build a second copy of the exit with the thread options and libraries:

```
$ CC /INCLUDE_DIRECTORY=MQS_INCLUDE exitname.C
$ LINK /SHARE=SYS$SHARE:MYFORMAT exitname.OBJ,MYFORMAT/OPTIONS
```

Again, the contents of MYFORMAT.OPT vary depending on what platform you are working on:

On AXP:
```
SYS$SHARE:MQM_R/SHAREABLE
SYS$SHARE:CMA$OPEN_RTL.EXE/SHAREABLE
SYMBOL_VECTOR=(MQSTART=PROCEDURE)
```

On VAX:
```
SYS$SHARE:MQM_R/SHAREABLE
SYS$SHARE:CMA$OPEN_RTL.EXE/SHAREABLE
UNIVERSAL=MQSTART
```

MQSeries for Compaq Tru64 UNIX

The exit is a dynamically loaded object that must be written in C. To ensure that it can be loaded when required, link and copy the exit to the /usr/lib directory. Define a dummy MQStart() routine in the exit and specify MQStart as the entry
Channel-exit programs

point in the module. Figure 138 shows how to set up an entry to your program:

```c
#include <cmqc.h>
#include <cmqxc.h>

void MQStart() {} /* dummy entry point - for consistency only */
void MQENTRY ChannelExit ( PMQVOID pChannelExitParms,
PMQVOID pChannelDefinition,
PMQLONG pDataLength,
PMQLONG pAgentBufferSize,
PMQVOID pAgentBuffer,
PMQLONG pExitBufferLength,
PMQPTR pExitBufferAddr)
{
  ... Insert code here
}
```

Figure 138. Sample source code for a channel exit on Compaq Tru64 UNIX

Figure 139 shows the compiler and loader commands for channel-exit programs on Compaq Tru64 UNIX.

```
$ cc -stdl -c -I/opt/mqm/inc exit.c
$ ld -o exit exit.o -shared -L/opt/mqm/lib -lmqm -e MQStart -lc
$ cp exit /usr/lib
```

Figure 139. Sample compiler and loader commands for channel exits on Compaq Tru64 UNIX

WebSphere MQ for HP-UX

Note: Before you use an existing user exit for the first time on WebSphere MQ for HP-UX, you must recompile it to enable it to take advantage of thread-safe system calls. If your user exits use thread-unsafe system calls, you will need to modify them before using them on this platform.

The exit is a dynamically loaded object that must be written in C. To ensure that it can be loaded when required, specify the full path name in the DEFINE CHANNEL command or enter the path name in the ExitPath stanza of the QM.INI file. If the exit is on an HP-UX client, specify the path name in the ClientExitPath stanza of the MQS.INI file. The value in the ExitPath stanza of the QM.INI file or the ClientExitPath stanza of the MQS.INI file defaults to /var/mqm/exits. You can change this value or you can override it by specifying a full path name on the DEFINE CHANNEL command.

Define a dummy MQStart() routine in the exit and specify MQStart as the entry point in the module. Figure 140 on page 598 shows how to set up an entry to your program:
Channel-exit programs

```c
#include <cmqc.h>
#include <cmqxc.h>

void MQStart() {} /* dummy entry point - for consistency only */
void MQENTRY ChannelExit ( PMQVOID pChannelExitParms,
PMQVOID pChannelDefinition,
PMQLONG pDataLength,
PMQLONG pAgentBufferLength,
PMQVOID pAgentBuffer,
PMQLONG pExitBufferLength,
PMQPTR pExitBufferAddr)
{
    ... Insert code here
}
```

Figure 140. Sample source code for a channel exit on HP-UX

```bash
$ cc -c +z exit.c
$ ld -o exit exit.o +b : -c exit.exp +I MQStart -b
$ cp exit /usr/xmp/lib # (or wherever you require)
```

Figure 141. Sample compiler and loader commands for channel exits on HP-UX

**MQSeries for AT&T GIS UNIX**

The exit is a dynamically loaded object that must be written in C. Specify the full path name in the DEFINE CHANNEL command. Define a dummy MQStart() routine in the exit and specify MQStart as the entry point in the module. Figure 142 shows how to set up an entry to your program:

```c
#include <cmqc.h>
#include <cmqxc.h>

void MQStart() {} /* dummy entry point */
void MQENTRY ChannelExit ( PMQVOID pChannelExitParms,
PMQVOID pChannelDefinition,
PMQLONG pDataLength,
PMQLONG pAgentBufferLength,
PMQVOID pAgentBuffer,
PMQLONG pExitBufferLength,
PMQPTR pExitBufferAddr)
{
    ... Insert code here
}
```

Figure 142. Sample source code for a channel exit on AT&T GIS UNIX

```bash
$ cc -c +z exit.c
$ ld -o exit exit.o +b : -c exit.exp +I MQStart -b
$ cp exit /usr/xmp/lib # (or wherever you require)
```

Figure 143 on page 599 shows the compiler and loader commands for channel-exit programs on AT&T GIS UNIX.

12. This platform has become NCR UNIX SVR4 MP-RAS, R3.0

598  WebSphere MQ Intercommunication
WebSphere MQ for Solaris

Note: Before you use an existing user exit for the first time on WebSphere MQ for Solaris, you must recompile it to enable it to take advantage of thread-safe system calls. If your user exits use thread-unsafe system calls, you will need to modify them before using them on this platform. If you have DCE installed, your channel exits must be threaded with DCE threading. If you do not have DCE installed, your channel exits must be threaded with Posix V10 threading.

The exit is a dynamically loaded object that must be written in C. To ensure that it can be loaded when required, specify the full path name in the DEFINE CHANNEL command or enter the path name in the ExitPath stanza of the QM.INI file. If the exit is on a Solaris client, specify the path name in the ClientExitPath stanza of the MQS.INI file. The value in the ExitPath stanza of the QM.INI file or the ClientExitPath stanza of the MQS.INI file defaults to /var/mqm/exits. You can change this value or you can override it by specifying a full path name on the DEFINE CHANNEL command.

Define a dummy MQStart() routine in the exit and specify MQStart as the entry point in the module. Figure 144 shows how to set up an entry to your program:

```c
#include <cmqc.h>
#include <cmqxc.h>

void MQStart() {} /* dummy entry point */
void MQENTRY ChannelExit ( PMQVOID pChannelExitParms,
                           PMQVOID pChannelDefinition,
                           PMQLONG pDataLength,
                           PMQLONG pAgentBufferLength,
                           PMQVOID pAgentBuffer,
                           PMQLONG pExitBufferLength,
                           PMQPTR pExitBufferAddr)
{
  ... Insert code here
}
```

Figure 144. Sample source code for a channel exit on Solaris

Figure 145 shows the compiler and loader commands for channel-exit programs on Solaris.

```bash
$ cc -c PIC exit.c
$ ld -o exit -G exit.o
$ cp exit /usr/xmp/lib # (or wherever you require)
```

Figure 145. Sample compiler and loader commands for channel exits on Solaris
MQSeries for SINIX and DC/OSx

The exit is a dynamically loaded object that must be written in C. Specify the full path name in the DEFINE CHANNEL command. Define a dummy MQStart() routine in the exit and specify MQStart as the entry point in the module. Figure 146 shows how to set up an entry to your program:

```
#include <cmqc.h>
#include <cmqxc.h>

void MQStart() {} /* dummy entry point */
void MQENTRY ChannelExit ( PMQVOID pChannelExitParms,
                           PMQVOID pChannelDefinition,
                           PMQLONG pDataLength,
                           PMQLONG pAgentBufferLength,
                           PMQVOID pAgentBuffer,
                           PMQLONG pExitBufferLength,
                           PMQPTR pExitBufferAddr)
{
    ... Insert code here
}
```

Figure 146. Sample source code for a channel exit on SINIX and DC/OSx

For DC/OSx, version cd087 and later, append the following to the cc line:
-liconv -lresolv

For earlier versions of DC/OSx, append the following to the cc line:
-liconv

MQSeries for Compaq NonStop Kernel

MQSeries for Compaq NonStop Kernel supports a single, statically bound channel-exit program, whose entry point is MQ_CHANNEL_EXIT(). The exit must be written in C. MQSeries for Compaq NonStop Kernel provides a stub function for this exit that acts as a placeholder for user-supplied exit code. In the supplied stub function, the ExitResponse field in MQCXP (channel exit parameter structure) is set to MQXCC_CLOSE_CHANNEL, which causes the MCA to close the channel. No other fields in MQCXP are modified.

You replace the supplied stub function in the MCA executable images with your own user exit code using the Tandem BIND utility BEXITE. Only the Tandem Common Runtime Environment (CRE) interface for the WIDE memory model is supported.

In MQSeries for Compaq NonStop Kernel, there is a single entry point for all channel exits. In other MQSeries Version 5 products, there are entry points specific to each channel type and function. It is possible to use channel-exit programs written for other MQSeries Version 5 products by calling those programs from MQ_CHANNEL_EXIT(). To determine the type of exit being called, examine the
ExitId field of MQCXP, then extract the associated exit-program name from the
MsgExit, MsgRetryExit, ReceiveExit, SendExit, or SecurityExit field of MQCD.

The channel attributes that define the names of user exits are:
- Security exit name (SCYEXIT)
- Message-retry exit name (MREXIT)
- Message exit name (MSGEXIT)
- Send exit name (SENDEXIT)
- Receive exit name (RCVEXIT)

If these channel attributes are left blank, the channel user exit is not invoked. If any of the channel attributes is nonblank, the MQ_CHANNEL_EXIT() user exit program is invoked for the corresponding function. Note that the text-string value of the channel attribute is not used to determine the name of the user exit program, since only a single entry point, MQ_CHANNEL_EXIT(), is supported in MQSeries for Compaq NonStop Kernel. However, the values of these channel attributes are passed to MQ_CHANNEL_EXIT() in the MQCD (channel data) structure. The function of the channel exit (that is, whether the exit corresponds to a Message, Message-retry, Receive, Security or Send Exit) is passed to MQ_CHANNEL_EXIT() in the ExitId field of the MQCXP (Channel Exit Parameters) structure.

MQSeries for Compaq NonStop Kernel does not support the following channel attributes:
- CICS Profile Name
- Sequential delivery
- Target system identifier
- Transaction identifier
- Maximum transmission size

Building and using channel exit functions
Dynamically bound libraries are not supported by WebSphere MQ for Compaq NonStop Kernel. Channel exits (and data-conversion exits) are implemented by including statically bound stub functions in the WebSphere MQ libraries and executables which can be replaced using the REPLACE bind option.

A channel exit function must be written in C, must be called CHANNELEXIT (see sample MQSVCHX), and can be bound into the chosen executable (or library) using the TACL macro BEXITE.

Note: This procedure modifies the target executable. Therefore, you are recommended to make a backup copy of the target executable or library before using the macro.

The function CHANNELEXIT must handle each of the possible exit calls (security, message-retry, message, send, and receive). You may write your own functions to do this.

Use the TACL macro BCHXALL to bind the data conversion exit into all required WebSphere MQ processes. For example:
BCHXALL source-exit-file-or-library

Sample channel exit:
/*****************************/
/*
* Program name: MQSVCHXE */
/*
/*
/* Description: Sample C skeleton of a Channel Exit controlling function
/* Statement: Licensed Materials - Property of IBM
/*
/* 33H2205, 5622-908
/* 33H2267, 5765-623
/* 29H0990, 5697-176
/* (C) Copyright IBM Corp. 1994, 1995
/*
********************************************************************/

/**/

/**/

/* Function: */

/* MQSVCHXE is a sample C skeleton of a Channel Exit controlling function */
/* The function controls the calls to user-defined channel exits */
/* */
/* */
/* Once complete the code should be compiled into a loadable object. */
/* the name of the object should be the name of the format to be converted. Instructions on how to do this are contained in the README file in the current directory. */
/* */
/* */

/**/

/**/

/* MQSVFCEXE takes the parameters defined for a Data Conversion exit routine in the CMQXC.H header file. */
/**/ 
/**/ 
/**/ 
/**/

#include <cmqc.h> /* For MQI datatypes */
#include <cmqxc.h> /* For MQI exit-related definitions */
#include <amqsvmht.h> /* For sample macro definitions */

/**/ 

/**/ 

MQDATACONVEXIT CHANNELEXIT;

/**/ 

/**/

MQENTRY CHANNELEXIT(
    PMQVOID pChannelExitParms, /* Channel exit parameter block */
    PMQVOID pChannelDefinition, /* Channel definition */
    PMQLONG pDataLength, /* Length of data */
    PMQLONG pAgentBufferLength, /* Length of agent buffer */
    PMQVOID pAgentBuffer, /* Agent buffer */
    PMQLONG pExitBufferLength, /* Length of exit buffer */
    PMQPTR pExitBufferAddr) /* Address of exit buffer */
{ 

    PMQCXP pCEP = pChannelExitParms;
    PMQCD pCD = pChannelDefinition;
    MQLONG ExitId = pCEP->ExitId;
    MQLONG ExitReason = pCEP->ExitReason;

    pCEP->ExitResponse = MQXCC_OK ;

    /* Call the handling function according to the ExitId */
Channel-exit programs

/* By default, there are no exits. If there are, then */
/* this function will have been replaced by a bind */

switch (ExitId)
{
    case MQXT_CHANNEL_SEC_EXIT:
        if (strlen(pCD->SecurityExit) == 0)
        {
            pCEP->ExitResponse = MQXCC_SUPPRESS_FUNCTION ;
        }
        else
        {
            /* Call the security exit function here */
        }
        break;
    case MQXT_CHANNEL_MSG_EXIT:
        /* Call the channel message exit function here */
        if (strlen(pCD->MsgExit) == 0)
        {
            pCEP->ExitResponse = MQXCC_SUPPRESS_FUNCTION ;
        }
        else
        {
            /* Call the message exit function here */
        }
        break;
    case MQXT_CHANNEL_SEND_EXIT:
        /* Call the channel send exit function here */
        if (strlen(pCD->SendExit) == 0)
        {
            pCEP->ExitResponse = MQXCC_SUPPRESS_FUNCTION ;
        }
        else
        {
            /* Call the send exit function here */
        }
        break;
    case MQXT_CHANNEL_RCV_EXIT:
        /* Call the channel receive exit function here */
        if (strlen(pCD->ReceiveExit) == 0)
        {
            pCEP->ExitResponse = MQXCC_SUPPRESS_FUNCTION ;
        }
        else
        {
            /* Call the receive exit function here */
        }
        break;
    case MQXT_CHANNEL_MSG_RETRY_EXIT:
        /* Call the channel retry exit function here */
        if (strlen(pCD->MsgRetryExit) == 0)
        {
            pCEP->ExitResponse = MQXCC_SUPPRESS_FUNCTION ;
        }
        else
        {
            /* Call the message retry exit function here */
        }
        break;
    default:
    { /* if the exit isn't recognized, stop it from being called again */
        pCEP->ExitResponse = MQXCC_SUPPRESS_EXIT ;
        break;
    } break;
}
return;

/*************************************************************************/
Supplied channel-exit programs using DCE security services

WebSphere MQ for AIX, HP-UX, Linux, Solaris, and Windows, and MQSeries V5.1 for Compaq Tru64 UNIX, and OS/2 Warp supply channel-exit programs for the security exit, the message exit, and the send and receive exits. The WebSphere MQ client for Windows 95 supplies channel-exit programs for the security exit and the send and receive exits. These programs take advantage of the Distributed Computing Environment (DCE) security services and encryption facilities. Before using the supplied exit programs from a WebSphere MQ client for Windows 95, see the note under “How to use the DCE channel-exit programs” on page 607.

The programs are supplied in source and object format. You can use the objects as they stand, or can use the source as the basis for creating your own user-exit programs. You should bear in mind that whereas the objects are supplied as working programs, the source code does not include any provision for tracing or error handling. If you chose to modify and use the source code, you should add your own tracing and error-handling routines.

You cannot use the supplied channel-exit programmes on a queue manager that also has SSL parameters set. This applies to queue managers on AIX, HP-UX, and Solaris.

The object has two entry points:

**DCE_SEC_SCY_CHANNELEXIT**
For the security exit, which can be used to access authentication services.

**DCE_SEC_SRMR_CHANNELEXIT**
For the send, receive, and message exits, which can be used to access data encryption services.

What do the DCE channel-exit programs do?

The supplied channel-exit programs address the Distributed Computing Environment (DCE) considerations for security in the areas of data encryption, and of authentication of a partner system when establishing a session. For a particular channel, each exit program has an associated DCE principal (similar to a user ID). A connection between two exit programs is an association between the two principals.

A secure connection between two security exit programs, one for the sending MCA and one for the receiving MCA, is established after the underlying session has been established. The sequence of operations is as follows:

1. Each program is associated with a particular principal, for example due to an explicit DCE Login.
2. The program that initiates the secure connection, that is the first program to get control after the MCA session has been established, is known as the Context Initiator. The context initiator requests a secure connection with the named partner from the DCE security server and receives a token. The token (called token1 in Figure 148 on page 605) is sent, using the already established underlying session, to the partner program.
3. The partner program (known as the Context Acceptor) passes token1 to the DCE security server, which verifies that the Context Initiator is authentic. For mutual
authentication, as implemented by the supplied security exit, the DCE security server also generates a second token (called token2 in Figure 148), which the Context Acceptor returns to the Context Initiator using the underlying session.

4. The Context Initiator uses token2 to verify that the Context Acceptor is authentic.
   At this stage, if both applications are satisfied with the authenticity of the partner’s token, then the secure (authenticated) connection is established.

5. The token exchange described above establishes a Security Context for each security exit program. This context enables the subsequent send, receive, and message exits to encrypt and decrypt data passed on the connection.
   DCE Security provides an API to ‘seal’ and ‘unseal’ data and hence to selectively protect specified elements of a datastream. The supplied message, send, and receive exits encrypt and decrypt messages using these DCE Security API calls.

<table>
<thead>
<tr>
<th>NODE name1</th>
<th>Flow</th>
<th>NODE name2</th>
</tr>
</thead>
<tbody>
<tr>
<td>gss_acquire_cred(name1)</td>
<td>gss_init_sec_context</td>
<td>gss_accept_sec_context</td>
</tr>
<tr>
<td>gss_init_sec_context (name2)-&gt;token1</td>
<td>INIT_SEC(token1)</td>
<td>(token1)-&gt;token2</td>
</tr>
<tr>
<td>gss_init_sec_context (name2)</td>
<td>ACC_SEC(token2)</td>
<td></td>
</tr>
</tbody>
</table>

Above two flows can be repeated, if required by GSS. When satisfied, proceed to other data transfer.

Figure 148. Security exit flows

The encryption algorithm used by the send exit must match the decryption algorithm used by the receive exit. The supplied send, receive, and message exits use the gss_seal() and gss_unseal() calls to encrypt and decrypt data. The qop_req parameter on the gss_seal() call is set to GSS_C_QOP_DEFAULT. The encryption provided by DCE depends on the DCE product installed.

The send, receive, and message exits are all used for encryption. The difference is that the message exit encrypts only the content of the message, whereas the send and receive exits also encrypt the message headers. Therefore, the message exit offers slightly better performance but at the expense of unencrypted header data.

How do the DCE channel-exit programs work?

The supplied code implements a security exit and message, send, and receive exits. Note that the message exit does not encrypt the WebSphere MQ header. The security exit provides mutual (two-way) authentication. The message, send, and receive exits provide encryption facilities based on a key managed by the security context set up by the security exit. Therefore, the message, send, and receive exits will not work unless the security exit has been called previously.

The code interfaces to DCE through the DCE GSS API provided as part of OSF DCE 1.1. This API provides a superset of the standard GSS API calls as specified in Internet RFCs 1508 and 1509. Some DCE-specific GSS calls have been added to the API by OSF.
Channel-exit programs

The principal of a WebSphere MQ system that has a queue manager is the queue manager name.

A WebSphere MQ client does not have a queue manager. The principal used for a client is as follows:

- On Solaris, AIX, HP-UX, Windows 95, and Windows clients:
  - If the login user ID of the user who started the WebSphere MQ client application can be obtained and is defined as a principal to DCE, this user ID is used.
  - If the login user ID of the user who started the WebSphere MQ client application cannot be obtained or is not defined to DCE, and a DCE default login context exists, the DCE default credential is used.

  **Note:** When a principal logs in to DCE, a default login context is established. In this case the principal used in association with the DCE default credential is that of the principal logged in to DCE.

  - If the login user ID of the user who started the WebSphere MQ client application cannot be obtained or is not defined to DCE, and no DCE default login context exists, there is no principal name available and the security exit rejects the attempt to start the channel.

- On OS/2 clients, user IDs cannot be used as principals.
  - If a principal has logged in to DCE, the name of this logged in principal is used.
  - If a principal has not logged in to DCE, and a DCE default login context exists, the DCE default credential is used.
  - If a principal has not logged in to DCE, and no DCE default login context exists, there is no principal name available and the security exit rejects the attempt to start the channel.

It is important that queue manager names or user IDs that are to be used as DCE principals are syntactically acceptable to DCE; see your DCE documentation for information about valid DCE principal names. If the name is to be used only within the local cell directory, the only mismatch between the allowable characters in a queue manager name and the allowable characters in a principal name is that a principal name cannot contain a ‘/’. If there is any likelihood that the name will also need to be reflected in a global directory, you are recommended to restrict principal names to alphanumeric characters. As with any DCE principal, when you create it you must define it to the DCE security server and must also put an entry for it in the relevant keytable file. Therefore, when you delete a queue manager that is also a DCE principal you must remember to delete both its entries.

Remote queue manager names are transferred across a channel at channel initialization. When the security exit is called, if the remote WebSphere MQ system is not a client, the remote queue manager name (which is also the remote principal) is passed to the security exit in the WebSphere MQ MQCXP parameter list. The initiator exit uses the name provided. If the channel is being established between a WebSphere MQ client and a WebSphere MQ server, the client always initiates the first security flow. In all cases, the initiator exit’s remote principal name is a queue manager name.

The flows shown in [Figure 148 on page 605](#) occur to establish the security context. As a part of these flows the initiator’s principal is transferred to the acceptor.
Channel-exit programs

It is possible to establish multiple security contexts between the same pair of principals, and hence to allow parallel channels to use the security exit.

You can set up restricted channels. The system administrator supplies a value in the Channel Security Exit User Data when defining this end of the channel. The presence of this value causes the security exit to check the remote principal name. If this check shows a mismatch the channel is not established. Note that the remote principals (queue manager names and default DCE principals) may be longer than the 32 characters allowed in the Channel Security Exit User Data. Only the first 32 characters of the remote principal are considered significant.

If the MCA forms part of a WebSphere MQ server system connected to a client, the security exchange will have caused the client principal to flow to the server. If the value is valid with regard to the optional restricted-channel check and the MCAUserIdentifier variable is not already defined, the client principal is copied into the server’s MCAUserIdentifier variable. Note that client principals may be longer than the 12-character MCAUserIdentifier. Only the first 12 characters of such a remote principal are copied.

Thus the first 12 characters of the WebSphere MQ client’s DCE principal name can become the user identifier to be used by the server’s MCA for authorization for that client to access WebSphere MQ resources. The server system must be set up appropriately to allow this to work.

How to use the DCE channel-exit programs

Do not run the supplied DCE message exit in combination with the supplied DCE send and receive exits on the same channel.

To use the supplied channel-exit programs you need to install DCE and define some channels. For installation information, see the Quick Beginnings book for your platform:

- The WebSphere MQ for AIX, V5.3 Quick Beginnings book.
- The WebSphere MQ for HP-UX, V5.3 Quick Beginnings book.
- The WebSphere MQ for Solaris, V5.3 Quick Beginnings book.
- The MQSeries for OS/2 Warp, V5.1 Quick Beginnings book.
- The WebSphere MQ for Windows, V5.3 Quick Beginnings book.

Note: Using IBM DCE for Windows 95 V1, you cannot use the supplied DCE security exit from a Windows 95 client connected to a WebSphere MQ for HP-UX server or a WebSphere MQ for Solaris server. Nor can you use the supplied send and receive exits from a Windows 95 client when using IBM DCE for Windows 95 V1.

Setup for DCE

The supplied channel-exit programs are intended for use between systems operating within a single DCE cell. The setup of a DCE cell is described in the documentation provided with the DCE packages for the platforms incorporated in the cell. The exit programs operate the same way whether they are running on a system with a DCE security client installed or with a DCE security server installed.

Once the DCE cell has been configured, it is necessary to define the principals that the exit is going to use to DCE. DCE setup samples are provided on all the supported platforms. The samples are primarily intended for setting up DCE for
the DCE Names installable component. They also contain comments indicating how they can be modified to set up the DCE security principals instead of, or as well as, the Names principal.

Each DCE security principal has its own keytable. On UNIX systems that support DCE security, the keytable is a file within the directory /var/mqm/dce/keytabs. On OS/2 and Windows it is a file within the directory \MQM\DCE\KEYTABS, where MQM is the name of your work path.

When the supplied channel-exit programs are called for a particular principal, they look in a keytable file that has the same name as the principal itself. Therefore, the keytable file for a particular principal must have the same name as that principal.

The use of separate keytables for each principal is recommended in the OSF DCE literature. On systems that support file access controls (UNIX systems and Windows) keytable access should be limited to:

- Superuser/administrator: no restriction
- Other user IDs:
  - read only access, given only to the user IDs under which the processes that call the security exits run, and only to the relevant keytables.

In the case of queue manager WebSphere MQ systems, the processes that interface to the security exits at the sending end of the channel are runmqch1 (and runmqchi on OS/2 and Windows NT). amqcrsta, amqcrs6a or runmqslr interface to the security exits at the receiving end of the channel. On most systems these all run under the mqm user ID; in this case, non-supervisor/administrator access to the keytables relating to queue manager principals should be restricted to read access for the mqm user ID.

On client systems the user ID under which the security exit is called is the user ID under which the client application runs (often the login user ID of the user of the client system). Again, non-supervisor/administrator access to the relevant keytable should be restricted to read access by that user ID only.

**The supplied exit code**
The supplied exit code is in two formats: object and source.

**Object:** The object is called amqrdsc0 on UNIX systems and amqrdsc0.DLL on OS/2, and Windows. It is installed as a standard part of the WebSphere MQ product for your platform and is loaded as a standard user exit. If you wish to run the supplied security channel exit to make use of authentication services then in your definition of the channel, specify:

```
SCYEXIT('<path>amqrdsc0(DCE_SEC_SCY_CHANNELEXIT)')
```

If you also wish to use the message exit to support data encryption, then in your definition of the channel, specify:

```
MSGEXIT('<path>amqrdsc0(DCE_SEC_SRM_CHANNELEXIT)')
```

Or you can use the send and receive exits to support data encryption by specifying the following in your definition of the channel:

```
SENDEXIT('<path>amqrdsc0(DCE_SEC_SRM_CHANNELEXIT)')
RCVEXIT('<path>amqrdsc0(DCE_SEC_SRM_CHANNELEXIT)')
```

<path> is the path to the directory containing the exit.
Channel-exit programs

See page 587 through page 600 for information about how to call user exits on the platform you are using.

Source: The exit source file is called amqsdsc0.c. It can be found in $mqmtop$/samp on UNIX systems and in $bootdrive>:\mqm\tools\c\samples$ on OS/2, and Windows. If you choose to modify the source versions, rather than running the objects as they stand, you will need to recompile the modified source. It is compiled and linked in the same way as any other channel exit for the platform concerned, except that DCE headers need to be accessed at compile time, and the DCE libraries, together with any recommended associated libraries, need to be accessed at link time. Refer to the documentation for the DCE product for the platform you are using, to find out about the DCE and associated libraries.

OS/2

```bash
icc /DIBMOS2 /DINTEL80x86 /Fe amqsdsc0.dll /I \
c:\mqclient\tools\c\include /I \
c:\ibmcpp\include /I c:\opt\dce\local\include\dce \
W3 /Sa /Ge- /Gm+ amqsdsc0.c amqsdsc0.def dceos2.lib
```

Using the following definition file:

```bash
LIBRARY AMQSDSC0
PROTMODE
DESCRIPTION 'DCE Security Exit'
CODE SHARED LOADONCALL
DATA NONSHARED MULTIPLE
HEAPSIZE 4096
STACKSIZE 8192
EXPORTS
  DCE_SEC_SCY_CHANNELEXIT
  DCE_SEC_SRM_CHANNELEXIT
```

Solaris

```bash
cc -I/opt/dce/share/include/dce \
  -I/opt/mqm/inc -KPIC -c amqsdsc0.c
```

followed by:

```bash
1d -G -L/opt/dce/share/usr/lib -ldce amqsdsc0.o -o srm
```

HP-UX

```bash
cc -D_HPUX_SOURCE -Dhpux -DICOL -D_REENTRANT \ 
-Dsigaction=cma_sigaction +ESlit +DA1.0 -c +z \ 
amqsdsc0.c -I /opt/mqm/include -I /opt/dce/include/dce \ 
-Aa && ld -o amqsdsc0 amqsdsc0.o -z +b : -b +I MQStart \ 
-ldce -1mqm_r -1ndbm -1M -1c_r
```

Windows 95, Windows 98, and Windows NT

```bash
c:\msdevstd\bin\cl /DAMQ_PC /VERBOSE /LD /MT \ 
/IC:\msdevstd\include /ID:\MQCLIENT\TOOLS\C\INCLUDE \ 
/IC:\OPT\DIGITAL\DCE\INCLUDE\DCE amqsdsc0.c \ 
-1link /DLL /EXPORT:DCE_SEC_SCY_CHANNELEXIT \ 
/EXPORT:DCE_SEC_SRM_CHANNELEXIT /STACK:8192 libdce.lib \ 
advapi32.lib libcmt.lib
```

AIX

```bash
x1C_r -c /usr/mqm/samp/amqsdsc0.c -I/usr/include/dce 
1d -e MQStart -bnoquiet -o amqsdsc0 amqsdsc0.o \ 
-L/usr/lib/dce -T512 -H512 -ldce -bE:amqsdsc0.exp \ 
-1pthread -1c_r -1iconv -1s
```
Using DCE channel exits with the runmqslr listener program
On WebSphere MQ for Windows, the exit dll name must be amqrdsc0.dll or amqsdscc0.dll.

SSPI security exit

WebSphere MQ for Windows supplies a security exit for both the WebSphere MQ client and the WebSphere MQ server. This is a channel-exit program that provides authentication for WebSphere MQ channels by using the Security Services Programming Interface (SSPI). The SSPI provides the integrated security facilities of Windows.

The security packages are loaded from either security.dll or secur32.dll. These DLLs are supplied with your operating system.

One-way authentication is provided on Windows, using NTLM authentication services. Two way authentication is provided on Windows 2000, using Kerberos authentication services.

The security exit program is supplied in source and object format. You can use the object code as it is, or you can use the source code as a starting point to create your own user-exit programs. For more information on using the object or source code of the SSPI security exit, see WebSphere MQ Application Programming Guide.
Chapter 46. Channel-exit calls and data structures

This chapter provides reference information about the special WebSphere MQ calls and data structures used when writing channel exit programs. This is product-sensitive programming interface information. You can write WebSphere MQ user exits in the following programming languages:

<table>
<thead>
<tr>
<th>Platform</th>
<th>Programming languages</th>
</tr>
</thead>
<tbody>
<tr>
<td>WebSphere MQ for z/OS without CICS</td>
<td>Assembler and C (which must conform to the C system programming environment for system exits, described in the OS/390 C/C++ Programming Guide.)</td>
</tr>
<tr>
<td>WebSphere MQ for z/OS using CICS</td>
<td>Assembler, C, COBOL, and PL/I</td>
</tr>
<tr>
<td>WebSphere MQ for iSeries</td>
<td>C, COBOL, and RPG II</td>
</tr>
<tr>
<td>All other WebSphere MQ platforms</td>
<td>C</td>
</tr>
</tbody>
</table>

You cannot write WebSphere MQ user exits in TAL or Visual Basic. However, a declaration for the MQCD structure is provided in Visual Basic for use on the MQCONNX call from an MQ client program.

In a number of cases, parameters are arrays or character strings whose size is not fixed. For these, a lowercase “n” is used to represent a numeric constant. When the declaration for that parameter is coded, the “n” must be replaced by the numeric value required. For further information about the conventions used in these descriptions, see the WebSphere MQ Application Programming Reference book.

The calls are:
- “MQ_CHANNEL_EXIT – Channel exit” on page 614
- “MQ_CHANNEL_AUTO_DEF_EXIT – Channel auto-definition exit” on page 619
- “MQ_TRANSPORT_EXIT – Transport retry exit” on page 622
- “MQXWAIT – Wait in exit” on page 624

The data structures are:
- “MQCD – Channel definition” on page 626
- “MQCXP – Channel exit parameter” on page 666
- “MQTXP – Transport exit parameter” on page 681
- “MQXWD – Exit wait descriptor” on page 685

Note: Channel exit programs are not supported on DOS or VSE/ESA.
Data definition files

The data definition files supplied with the products for each of the supported programming languages are shown in the following tables.

Table 54. C header files

<table>
<thead>
<tr>
<th>File name</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>CMQC</td>
<td>Function prototypes, data types, and named constants for the main MQI</td>
</tr>
<tr>
<td>CMQXC</td>
<td>Function prototypes, data types, and named constants for channel exits</td>
</tr>
</tbody>
</table>

Table 55. COBOL COPY files

<table>
<thead>
<tr>
<th>File name (with initial values)</th>
<th>File name (without initial values)</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>CMQV</td>
<td>–</td>
<td>Named constants for main MQI</td>
</tr>
<tr>
<td>CMQXV</td>
<td>–</td>
<td>Named constants for channel exits</td>
</tr>
<tr>
<td>CMQCDV</td>
<td>CMQCDL</td>
<td>Channel definition structure</td>
</tr>
<tr>
<td>CMQCXPV</td>
<td>CMQCXPL</td>
<td>Channel exit parameter structure</td>
</tr>
<tr>
<td>CMQDLHV</td>
<td>CMQDLHL</td>
<td>Dead letter header structure</td>
</tr>
<tr>
<td>CMQXPV</td>
<td>CMQXPL</td>
<td>CICS API exit parameter structure</td>
</tr>
<tr>
<td>CMQXQHV</td>
<td>CMQXQHL</td>
<td>Transmission queue header structure</td>
</tr>
</tbody>
</table>

Table 56. PL/I INCLUDE files

<table>
<thead>
<tr>
<th>File name</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>CMQP</td>
<td>Structures and named constants for main MQI</td>
</tr>
<tr>
<td>CMQEPP</td>
<td>Entry-point declarations for main MQI</td>
</tr>
<tr>
<td>CMQXP</td>
<td>Structures and named constants for channel exits</td>
</tr>
</tbody>
</table>

Table 57. RPG (ILE) COPY files

<table>
<thead>
<tr>
<th>File name (with initial values)</th>
<th>File name (without initial values)</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>CMQG</td>
<td>–</td>
<td>Named constants for main MQI</td>
</tr>
<tr>
<td>CMQXG</td>
<td>–</td>
<td>Named constants for channel exits</td>
</tr>
<tr>
<td>CMQCDG</td>
<td>CMQCDH</td>
<td>Channel definition structure</td>
</tr>
<tr>
<td>CMQCXPG</td>
<td>CMQCXPH</td>
<td>Channel exit parameter structure</td>
</tr>
<tr>
<td>CMQDLHG</td>
<td>CMQDLHH</td>
<td>Dead letter header structure</td>
</tr>
<tr>
<td>CMQXQHG</td>
<td>CMQXQHH</td>
<td>Transmission queue header structure</td>
</tr>
</tbody>
</table>

Table 58. RPG (OPM) COPY files

<table>
<thead>
<tr>
<th>File name</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>CMQXR</td>
<td>Named constants for channel exits</td>
</tr>
<tr>
<td>CMQCDR</td>
<td>Channel definition structure</td>
</tr>
</tbody>
</table>
### Table 58. RPG (OPM) COPY files (continued)

<table>
<thead>
<tr>
<th>File name</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>CMQCXPR</td>
<td>Channel exit parameter structure</td>
</tr>
<tr>
<td>CMQDLHR</td>
<td>Dead letter header structure</td>
</tr>
<tr>
<td>CMQXQHR</td>
<td>Transmission queue header structure</td>
</tr>
</tbody>
</table>

### Table 59. System/390® assembler macros

<table>
<thead>
<tr>
<th>File name</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>CMQA</td>
<td>Named constants (&quot;equates&quot;) for main MQI</td>
</tr>
<tr>
<td>CMQXA</td>
<td>Named constants for channel exits</td>
</tr>
<tr>
<td>CMQVERA</td>
<td>Structure version control</td>
</tr>
<tr>
<td>CMQCDA</td>
<td>Channel definition structure</td>
</tr>
<tr>
<td>CMQCXPA</td>
<td>Channel exit parameter structure</td>
</tr>
<tr>
<td>CMQDLHA</td>
<td>Dead letter header structure</td>
</tr>
<tr>
<td>CMQXPA</td>
<td>CICS API exit parameter structure</td>
</tr>
<tr>
<td>CMQXQHA</td>
<td>Transmission queue header structure</td>
</tr>
</tbody>
</table>

For a list of the complete set of header files for the product, see the [WebSphere MQ Application Programming Guide](#).
MQ_CHANNEL_EXIT

MQ_CHANNEL_EXIT – Channel exit

This call definition is provided solely to describe the parameters that are passed to each of the channel exits called by the Message Channel Agent. No entry point called MQ_CHANNEL_EXIT is actually provided by the queue manager; the name MQ_CHANNEL_EXIT is of no special significance since the names of the channel exits are provided in the channel definition MQCD.

This definition is part of the WebSphere MQ Security Enabling Interface (SEI), which is one of the WebSphere MQ framework interfaces.

There are five types of channel exit:
- Channel security exit
- Channel message exit
- Channel send exit
- Channel receive exit
- Channel message-retry exit

The parameters are similar for each type of exit, and the description given here applies to all of them, except where specifically noted.

Syntax

MQ_CHANNEL_EXIT (ChannelExitParms, ChannelDefinition, DataLength, AgentBufferLength, AgentBuffer, ExitBufferLength, ExitBufferAddr)

Parameters

The MQ_CHANNEL_EXIT call has the following parameters.

ChannelExitParms (MQCXP) – input/output
Channel exit parameter block.

This structure contains additional information relating to the invocation of the exit. The exit sets information in this structure to indicate how the MCA should proceed.

ChannelDefinition (MQCD) – input/output
Channel definition.

This structure contains parameters set by the administrator to control the behavior of the channel.

DataLength (MQLONG) – input/output
Length of data.

When the exit is invoked, this contains the length of data in the AgentBuffer parameter. The exit must set this to the length of the data in either the AgentBuffer or the ExitBufferAddr (as determined by the ExitResponse2 field in the ChannelExitParms parameter) that is to proceed.

The data depends on the type of exit:
- For a channel security exit, when the exit is invoked this contains the length of any security message in the AgentBuffer field, if ExitReason is MQXR_SEC_MSG. It is zero if there is no message. The exit must set this field to the length of any security message to be sent to its partner if it sets
**ExitResponse** to MQXCC_SEND_SEC_MSG or MQXCC_SEND_AND_REQUEST_SEC_MSG. The message data is in either AgentBuffer or ExitBufferAddr.

The content of security messages is the sole responsibility of the security exits.

- For a channel message exit, when the exit is invoked this contains the length of the message (including the transmission queue header). The exit must set this field to the length of the message in either AgentBuffer or ExitBufferAddr that is to proceed.
- For a channel send or channel receive exit, when the exit is invoked this contains the length of the transmission. The exit must set this field to the length of the transmission in either AgentBuffer or ExitBufferAddr that is to proceed.

If a security exit sends a message, and there is no security exit at the other end of the channel, or the other end sets an ExitResponse of MQXCC_OK, the initiating exit is re-invoked with MQXR_SEC_MSG and a null response (DataLength=0).

**AgentBufferLength (MQLONG) – input**
Length of agent buffer.

This can be greater than DataLength on invocation.

For channel message, send, and receive exits, any unused space on invocation can be used by the exit to expand the data in place. If this is done, the DataLength parameter must be set appropriately by the exit.

In the C programming language, this parameter is passed by address.

**AgentBuffer (MQBYTE×AgentBufferLength) – input/output**
Agent buffer.

The contents of this depend upon the exit type:

- For a channel security exit, on invocation of the exit it contains a security message if ExitReason is MQXR_SEC_MSG. If the exit wishes to send a security message back, it can either use this buffer or its own buffer (ExitBufferAddr).
- For a channel message exit, on invocation of the exit this contains:
  - The transmission queue header (MQXQH), which includes the message descriptor (which itself contains the context information for the message), immediately followed by
  - The message data

If the message is to proceed, the exit can do one of the following:

- Leave the contents of the buffer untouched
- Modify the contents in place (returning the new length of the data in DataLength; this must not be greater then AgentBufferLength)
- Copy the contents to the ExitBufferAddr, making any required changes

Any changes that the exit makes to the transmission queue header are not checked; however, erroneous modifications may mean that the message cannot be put at the destination.

- For a channel send or receive exit, on invocation of the exit this contains the transmission data. The exit can do one of the following:
  - Leave the contents of the buffer untouched
  - Modify the contents in place (returning the new length of the data in DataLength; this must not be greater then AgentBufferLength)
  - Copy the contents to the ExitBufferAddr, making any required changes
MQ_CHANNEL_EXIT — AgentBuffer parameter

Note that the first 8 bytes of the data must not be changed by the exit.

**ExitBufferLength (MQLONG) — input/output**
Length of exit buffer.

On the first invocation of the exit, this is set to zero. Thereafter whatever value is passed back by the exit, on each invocation, is presented to the exit next time it is invoked. The value is not used by the MCA (except in WebSphere MQ for z/OS using CICS for distributed queue management, where a check is made that DataLength does not exceed ExitBufferLength, if the exit is returning data in ExitBufferAddr).

**Note:** This parameter should not be used by exits written in programming languages which do not support the pointer data type.

**ExitBufferAddr (MQPTR) — input/output**
Address of exit buffer.

This is a pointer to the address of a buffer of storage managed by the exit, where it can choose to return message or transmission data (depending upon the type of exit) to the agent if the agent’s buffer is or may not be large enough, or if it is more convenient for the exit to do so.

On the first invocation of the exit, the address passed to the exit is null. Thereafter whatever address is passed back by the exit, on each invocation, is presented to the exit the next time it is invoked.

**Note:** This parameter should not be used by exits written in programming languages that do not support the pointer data type.

**Usage notes**

1. The function performed by the channel exit is defined by the provider of the exit. The exit, however, must conform to the rules defined here and in the associated control block, the MQCXP.

2. The ChannelDefinition parameter passed to the channel exit may be one of several versions. See the Version field in the MQCD structure for more information.

3. If the channel exit receives an MQCD structure with the Version field set to a value greater than MQCD_VERSION_1, the exit should use the ConnectionName field in MQCD, in preference to the ShortConnectionName field.

4. In general, channel exits are allowed to change the length of message data. This may arise as a result of the exit adding data to the message, or removing data from the message, or compressing or encrypting the message. However, special restrictions apply if the message is a segment that contains only part of a logical message. In particular, there must be no net change in the length of the message as a result of the actions of complementary sending and receiving exits.

For example, it is permissible for a sending exit to shorten the message by compressing it, but the complementary receiving exit must restore the original length of the message by decompressing it, so that there is no net change in the length of the message.

This restriction arises because changing the length of a segment would cause the offsets of later segments in the message to be incorrect, and this would inhibit the queue manager’s ability to recognize that the segments formed a complete logical message.
C invocation

```
exitname (&ChannelExitParms, &ChannelDefinition, 
        &DataLength, &AgentBufferLength, AgentBuffer, 
        &ExitBufferLength, &ExitBufferAddr);
```

The parameters passed to the exit are declared as follows:

- MQCXP ChannelExitParms; /* Channel exit parameter block */
- MQCD ChannelDefinition; /* Channel definition */
- MQLONG DataLength; /* Length of data */
- MQLONG AgentBufferLength; /* Length of agent buffer */
- MQBYTE AgentBuffer[n]; /* Agent buffer */
- MQLONG ExitBufferLength; /* Length of exit buffer */
- MQPTR ExitBufferAddr; /* Address of exit buffer */

COBOL invocation

```
CALL 'exitname' USING CHANNELEXITPARMS, CHANNELDEFINITION, 
       DATALENGTH, AGENTBUFFERLENGTH, AGENTBUFFER, 
       EXITBUFFERLENGTH, EXITBUFFERADDR.
```

The parameters passed to the exit are declared as follows:

- ** Channel exit parameter block
  01 CHANNELEXITPARMS.
    COPY CMQCXPV.
- ** Channel definition
  01 CHANNELDEFINITION.
    COPY CMQCDV.
- ** Length of data
  01 DATALENGTH PIC S9(9) BINARY.
- ** Length of agent buffer
  01 AGENTBUFFERLENGTH PIC S9(9) BINARY.
- ** Agent buffer
  01 AGENTBUFFER PIC X(n).
- ** Length of exit buffer
  01 EXITBUFFERLENGTH PIC S9(9) BINARY.
- ** Address of exit buffer
  01 EXITBUFFERADDR POINTER.

PL/I invocation

```
call exitname (ChannelExitParms, ChannelDefinition, DataLength, 
               AgentBufferLength, AgentBuffer, ExitBufferLength, 
               ExitBufferAddr);
```

The parameters passed to the exit are declared as follows:

- dcl ChannelExitParms like MQCXP; /* Channel exit parameter 
  block */
- dcl ChannelDefinition like MQCD; /* Channel definition */
- dcl DataLength fixed bin(31); /* Length of data */
- dcl AgentBufferLength fixed bin(31); /* Length of agent buffer */
- dcl AgentBuffer char(n); /* Agent buffer */
- dcl ExitBufferLength fixed bin(31); /* Length of exit buffer */
- dcl ExitBufferAddr pointer; /* Address of exit buffer */

RPG invocation (ILE)

```
C*..1........2........3........4........5........6........7..
C         CALLP  exitname(MQCXP : MQCD : DATLEN : 
C                              ABUFL : ABUF : EBUFL : 
C                              EBUF)
```

The prototype definition for the call is:
Language invocations

D*...1.........2...........3...........4...........5...........6...........7..
Dexitname PR EXTPROC('exitname')
D* Channel exit parameter block
D MQCX 160A
D* Channel definition
D MQCD 1328A
D* Length of data
D DATLEN 10I 0
D* Length of agent buffer
D ABUFL 10I 0
D* Agent buffer
D ABUF * VALUE
D* Length of exit buffer
D EBUFL 10I 0
D* Address of exit buffer
D EBUF *

C*..1....:....2....:....3....:....4....:....5....:....6....:....7..
C CALL 'exitname'
C* Channel exit parameter block
C PARM MQCX
C* Channel definition
C PARM MQCD
C* Length of data
C PARM DATLEN 90
C* Length of agent buffer
C PARM ABUFL 90
C* Agent buffer
C PARM ABUF n
C* Length of exit buffer
C PARM EBUFL 90
C* Address of exit buffer
C PARM EBUF 16

RPG invocation (OPM)

C*..1....:....2....:....3....:....4....:....5....:....6....:....7..
C CALL 'exitname'
C* Channel exit parameter block
C PARM MQCX
C* Channel definition
C PARM MQCD
C* Length of data
C PARM DATLEN 90
C* Length of agent buffer
C PARM ABUFL 90
C* Agent buffer
C PARM ABUF n
C* Length of exit buffer
C PARM EBUFL 90
C* Address of exit buffer
C PARM EBUF 16

Declare the structure parameters as follows:

I*..1.........2...........3...........4...........5...........6...........7..
I* Channel exit parameter block
I MQCXPR DS
I/COPY CMQCPXR
I* Channel definition
I MQCDR DS
I/COPY CMQCDR

System/390 assembler invocation

CALL EXITNAME,(CHANNELEXITPARMS,CHANNELDEFINITION,DATALENGTH, X AGENTBUFFERLENGTH,AGENTBUFFER,EXITBUFFERLENGTH, X EXITBUFFERADDR)

The parameters passed to the exit are declared as follows:

CHANNELEXITPARMS CMQCPXPA , Channel exit parameter block
CHANNELDEFINITION CMQCDR , Channel definition
DATALENGTH DS F Length of data
AGENTBUFFERLENGTH DS F Length of agent buffer
AGENTBUFFER DS CL(n) Agent buffer
EXITBUFFERLENGTH DS F Length of exit buffer
EXITBUFFERADDR DS F Address of exit buffer
MQ_CHANNEL_AUTO_DEF_EXIT – Channel auto-definition exit

This call definition is provided solely to describe the parameters that are passed to the channel auto-definition exit called by the Message Channel Agent. No entry point called MQ_CHANNEL_AUTO_DEF_EXIT is actually provided by the queue manager; the name MQ_CHANNEL_AUTO_DEF_EXIT is of no special significance because the names of the auto-definition exits are provided in the queue manager.

The MQ_CHANNEL_AUTO_DEF_EXIT call definition is part of the WebSphere MQ Security Enabling Interface (SEI), which is one of the WebSphere MQ framework interfaces.

This exit is supported in the following environments: AIX, HP-UX, Linux, z/OS, OS/2, iSeries, Solaris, Windows.

Syntax

MQ_CHANNEL_AUTO_DEF_EXIT (ChannelExitParms, ChannelDefinition)

Parameters

The MQ_CHANNEL_AUTO_DEF_EXIT call has the following parameters.

ChannelExitParms (MQCXP) – input/output
Channel exit parameter block.

This structure contains additional information relating to the invocation of the exit. The exit sets information in this structure to indicate how the MCA should proceed.

ChannelDefinition (MQCD) – input/output
Channel definition.

This structure contains parameters set by the administrator to control the behavior of channels which are created automatically. The exit sets information in this structure to modify the default behavior set by the administrator.

The MQCD fields listed below must not be altered by the exit:

- ChannelName
- ChannelType
- StructLength
- Version

If other fields are changed, the value set by the exit must be valid. If the value is not valid, an error message is written to the error log file or displayed on the console (as appropriate to the environment).

Usage notes

1. The function performed by the channel exit is defined by the provider of the exit. The exit, however, must conform to the rules defined here and in the associated control block, the MQCXP.

2. The ChannelExitParms parameter passed to the channel auto-definition exit is an MQCXP structure. The version of MQCXP passed depends on the environment in which the exit is running; see the description of the Version field in “MQCXP – Channel exit parameter” on page 666 for details.
3. The `ChannelDefinition` parameter passed to the channel auto-definition exit is an MQCD structure. The version of MQCD passed depends on the environment in which the exit is running; see the description of the `Version` field in "MQCD – Channel definition" on page 626 for details.

**C invocation**

```c
exitname (&ChannelExitParms, &ChannelDefinition);
```

The parameters passed to the exit are declared as follows:

```c
MQCXP ChannelExitParms; /* Channel exit parameter block */
MQCD ChannelDefinition; /* Channel definition */
```

**COBOL invocation**

```cobol
CALL 'exitname' USING CHANNELEXITPARMS, CHANNELDEFINITION.
```

The parameters passed to the exit are declared as follows:

```cobol
** Channel exit parameter block
  01 CHANNELEXITPARMS.
    COPY CMQCXPV.
  ** Channel definition
  01 CHANNELDEFINITION.
    COPY CMQCDV.
```

**RPG invocation (ILE)**

```rpg
CALLP exitname(MQCXP : MQCD)
```

The prototype definition for the call is:

```rpg
D exitname PR EXTPROC('exitname')
D* Channel exit parameter block
  D MQCXP 160A
D* Channel definition
  D MQCD 1328A
```

**RPG invocation (OPM)**

```rpg
CALL 'exitname'
```

```rpg
C* Channel exit parameter block
  C PARM MQCXP
C* Channel definition
  C PARM MQCD
```

Declare the structure parameters as follows:

```rpg
I* Channel exit parameter block
  I/COPY CMQCXPR
I* Channel definition
  I/COPY CMQCDR
```

**System/390 assembler invocation**

```asm
CALL EXITNAME,(CHANNELEXITPARMS,CHANNELDEFINITION)
```

The parameters passed to the exit are declared as follows:
Language invocations

CHANELEXITPARMS CMQCXA , Channel exit parameter block
CHANNELDEFINITION CMQCD , Channel definition
MQ_TRANSPORT_EXIT

MQ_TRANSPORT_EXIT – Transport retry exit

This call definition is provided solely to describe the parameters that are passed to the transport retry exit called by the message channel agent (MCA). No entry point called MQ_TRANSPORT_EXIT is actually provided by the MCA; the name MQ_TRANSPORT_EXIT is of no special significance because the name of the transport retry exit is provided by the queue-manager’s configuration file.

This exit is supported in the following environments: AIX.

Syntax

MQ_TRANSPORT_EXIT (ExitParms, DestAddressLength, DestAddress)

Parameters

The MQ_TRANSPORT_EXIT call has the following parameters.

ExitParms (MQTXP) – input/output
Exit parameter block.

This structure contains information relating to the invocation of the exit. The exit sets information in this structure to indicate how processing should continue.

DestAddressLength (MQLONG) – input
Length in bytes of destination IP address.

This is the length of the destination IP address DestAddress. The value is always greater than zero.

DestAddress (MQCHAR×DestAddressLength) – input
Destination IP address.

This is the IP address of the destination. Its length is given by the DestAddressLength parameter.

Usage notes

1. The function performed by the exit is defined by the provider of the exit. The exit, however, must conform to the rules defined in the associated control block MQTXP.
2. The transport retry exit allows a channel to be paused based on criteria that are external to WebSphere MQ.

If configured, the exit is called before each attempt to resend a failing data packet. When called, the exit can wait based on some external criterion, and not return control to the MCA until the exit decides that the resend of the data packet is likely to succeed. If the exit decides that transmission should be discontinued, the exit can instruct the MCA to close the channel.

C invocation

exitname (&ExitParms, DestAddressLength, DestAddress);

The parameters passed to the exit are declared as follows:
C invocation

MQTXP ExitParms; /* Exit parameter block */
MQLONG DestAddressLength; /* Length in bytes of destination IP address */
MQCHAR DestAddress[n]; /* Destination IP address */
MQXWAIT – Wait in exit

The MQXWAIT call waits for an event to occur. It can be used only from a channel exit on z/OS when not using CICS.

Syntax

MQXWAIT (Hconn, WaitDesc, CompCode, Reason)

Parameters

The MQXWAIT call has the following parameters.

**Hconn (MQHCONN) – input**

Connection handle.

This handle represents the connection to the queue manager. The value of Hconn was returned by a previous MQCONN call issued in the same or earlier invocation of the exit.

**WaitDesc (MQXWD) – input/output**

Wait descriptor.

This describes the event to wait for. See “MQXWD – Exit wait descriptor” on page 685 for details of the fields in this structure.

**CompCode (MQLONG) – output**

Completion code.

It is one of the following:

MQCC_OK

Successful completion.

MQCC_FAILED

Call failed.

**Reason (MQLONG) – output**

Reason code qualifying CompCode.

If CompCode is MQCC_OK:

MQRC_NONE

(0, X’000’) No reason to report.

If CompCode is MQCC_FAILED:

MQRC_ADAPTER_NOT_AVAILABLE

(2204, X’89C’) Adapter not available.

MQRC_OPTIONS_ERROR

(2046, X’7FE’) Options not valid or not consistent.

MQRC_XWAIT_CANCELED

(2107, X’83B’) MQXWAIT call canceled.

MQRC_XWAIT_ERROR

(2108, X’83C’) Invocation of MQXWAIT call not valid.

For more information on these reason codes, see the WebSphere MQ Application Programming Reference.
C invocation

MQXWAIT (Hconn, &WaitDesc, &CompCode, &Reason);

Declare the parameters as follows:

MQHCONN  Hconn; /* Connection handle */
MQXWD    WaitDesc; /* Wait descriptor */
MQLONG   CompCode; /* Completion code */
MQLONG   Reason; /* Reason code qualifying CompCode */

System/390 assembler invocation

CALL MQXWAIT,(HCONN,WAITDESC,COMPCODE,REASON)

Declare the parameters as follows:

HCONN   DS   F Connection handle
WAITDESC CMQXWDA  , Wait descriptor
COMPCODE DS   F Completion code
REASON   DS   F Reason code qualifying COMPCODE
MQCD – Channel definition

The following table summarizes the fields in the structure.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>ChannelName</td>
<td>Channel definition name</td>
<td>628</td>
</tr>
<tr>
<td>Version</td>
<td>Structure version number</td>
<td>629</td>
</tr>
<tr>
<td>ChannelType</td>
<td>Channel type</td>
<td>630</td>
</tr>
<tr>
<td>TransportType</td>
<td>Transport type</td>
<td>630</td>
</tr>
<tr>
<td>Desc</td>
<td>Channel description</td>
<td>631</td>
</tr>
<tr>
<td>QMgName</td>
<td>Queue manager name</td>
<td>631</td>
</tr>
<tr>
<td>XmitQName</td>
<td>Transmission queue name</td>
<td>631</td>
</tr>
<tr>
<td>ShortConnectionName</td>
<td>First 20 bytes of connection name</td>
<td>631</td>
</tr>
<tr>
<td>MCAName</td>
<td>Reserved</td>
<td>632</td>
</tr>
<tr>
<td>ModeName</td>
<td>LU 6.2 mode name</td>
<td>632</td>
</tr>
<tr>
<td>TpName</td>
<td>LU 6.2 transaction program name</td>
<td>632</td>
</tr>
<tr>
<td>BatchSize</td>
<td>Batch size</td>
<td>632</td>
</tr>
<tr>
<td>DiscInterval</td>
<td>Disconnect interval</td>
<td>632</td>
</tr>
<tr>
<td>ShortRetryCount</td>
<td>Short retry count</td>
<td>633</td>
</tr>
<tr>
<td>ShortRetryInterval</td>
<td>Short retry wait interval</td>
<td>633</td>
</tr>
<tr>
<td>LongRetryCount</td>
<td>Long retry count</td>
<td>633</td>
</tr>
<tr>
<td>LongRetryInterval</td>
<td>Long retry wait interval</td>
<td>633</td>
</tr>
<tr>
<td>SecurityExit</td>
<td>Channel security exit name</td>
<td>634</td>
</tr>
<tr>
<td>MsgExit</td>
<td>Channel message exit name</td>
<td>634</td>
</tr>
<tr>
<td>SendExit</td>
<td>Channel send exit name</td>
<td>634</td>
</tr>
<tr>
<td>ReceiveExit</td>
<td>Channel receive exit name</td>
<td>635</td>
</tr>
<tr>
<td>SeqNumberWrap</td>
<td>Highest allowable message sequence number</td>
<td>635</td>
</tr>
<tr>
<td>MaxMsgLength</td>
<td>Maximum message length</td>
<td>635</td>
</tr>
<tr>
<td>PutAuthority</td>
<td>Put authority</td>
<td>635</td>
</tr>
<tr>
<td>DataConversion</td>
<td>Data conversion</td>
<td>635</td>
</tr>
<tr>
<td>SecurityUserData</td>
<td>Channel security exit user data</td>
<td>636</td>
</tr>
<tr>
<td>MsgUserData</td>
<td>Channel message exit user data</td>
<td>636</td>
</tr>
<tr>
<td>SendUserData</td>
<td>Channel send exit user data</td>
<td>636</td>
</tr>
<tr>
<td>ReceiveUserData</td>
<td>Channel receive exit user data</td>
<td>637</td>
</tr>
</tbody>
</table>

**Note:** The remaining fields are not present if Version is less than MQCD\_VERSION\_2.

| UserIdentifier            | User identifier                                      | 637  |
| Password                  | Password                                              | 637  |
| MCAUserIdentifier         | First 12 bytes of MCA user identifier                | 637  |
| MCAType                   | Message channel agent type                            | 638  |
| ConnectionName            | Connection name                                      | 639  |
| RemoteUserIdentifier      | First 12 bytes of user identifier from partner       | 639  |
| RemotePassword            | Password from partner                                 | 640  |
Table 60. Fields in MQCD (continued)

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Note: The remaining fields are not present if Version is less than MQCD_VERSION_3.</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>MsgRetryExit</strong></td>
<td>Channel message retry exit name</td>
<td>640</td>
</tr>
<tr>
<td><strong>MsgRetryUserData</strong></td>
<td>Channel message retry exit user data</td>
<td>641</td>
</tr>
<tr>
<td><strong>MsgRetryCount</strong></td>
<td>Number of times MCA will try to put the message after the first attempt has failed</td>
<td>641</td>
</tr>
<tr>
<td><strong>MsgRetryInterval</strong></td>
<td>Minimum interval in milliseconds after which the open or put operation will be retried</td>
<td>642</td>
</tr>
<tr>
<td>Note: The remaining fields are not present if Version is less than MQCD_VERSION_4.</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>HeartbeatInterval</strong></td>
<td>Time in seconds between heartbeat flows</td>
<td>642</td>
</tr>
<tr>
<td><strong>BatchInterval</strong></td>
<td>Batch duration</td>
<td>643</td>
</tr>
<tr>
<td><strong>NonPersistentMsgSpeed</strong></td>
<td>Speed at which nonpersistent messages are sent</td>
<td>643</td>
</tr>
<tr>
<td><strong>StrucLength</strong></td>
<td>Length of MQCD structure</td>
<td>644</td>
</tr>
<tr>
<td><strong>ExitNameLength</strong></td>
<td>Length of exit name</td>
<td>644</td>
</tr>
<tr>
<td><strong>ExitDataLength</strong></td>
<td>Length of exit user data</td>
<td>644</td>
</tr>
<tr>
<td><strong>MsgExitsDefined</strong></td>
<td>Number of message exits defined</td>
<td>645</td>
</tr>
<tr>
<td><strong>SendExitsDefined</strong></td>
<td>Number of send exits defined</td>
<td>645</td>
</tr>
<tr>
<td><strong>ReceiveExitsDefined</strong></td>
<td>Number of receive exits defined</td>
<td>645</td>
</tr>
<tr>
<td><strong>MsgExitPtr</strong></td>
<td>Address of first MsgExit field</td>
<td>645</td>
</tr>
<tr>
<td><strong>MsgUserDataPtr</strong></td>
<td>Address of first MsgUserData field</td>
<td>645</td>
</tr>
<tr>
<td><strong>SendExitPtr</strong></td>
<td>Address of first SendExit field</td>
<td>646</td>
</tr>
<tr>
<td><strong>SendUserDataPtr</strong></td>
<td>Address of first SendUserData field</td>
<td>646</td>
</tr>
<tr>
<td><strong>ReceiveExitPtr</strong></td>
<td>Address of first ReceiveExit field</td>
<td>647</td>
</tr>
<tr>
<td><strong>ReceiveUserDataPtr</strong></td>
<td>Address of first ReceiveUserData field</td>
<td>647</td>
</tr>
<tr>
<td>Note: The remaining fields are not present if Version is less than MQCD_VERSION_5.</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>ClusterPtr</strong></td>
<td>Address of a list of clusters</td>
<td>648</td>
</tr>
<tr>
<td><strong>ClustersDefined</strong></td>
<td>Number of cluster records</td>
<td>648</td>
</tr>
<tr>
<td><strong>NetworkPriority</strong></td>
<td>Network priority</td>
<td>648</td>
</tr>
<tr>
<td>Note: The remaining fields are not present if Version is less than MQCD_VERSION_6.</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>LongMCAUserIdLength</strong></td>
<td>Length of long MCA user identifier</td>
<td>648</td>
</tr>
<tr>
<td><strong>LongRemoteUserIdLength</strong></td>
<td>Length of long remote user identifier</td>
<td>648</td>
</tr>
<tr>
<td><strong>LongMCAUserIdPtr</strong></td>
<td>Address of long MCA user identifier</td>
<td>649</td>
</tr>
<tr>
<td><strong>LongRemoteUserIdPtr</strong></td>
<td>Address of long remote user identifier</td>
<td>649</td>
</tr>
<tr>
<td><strong>MCASecurityId</strong></td>
<td>MCA security identifier</td>
<td>649</td>
</tr>
<tr>
<td><strong>RemoteSecurityId</strong></td>
<td>Remote security identifier</td>
<td>650</td>
</tr>
<tr>
<td>Note: The remaining fields are not present if Version is less than MQCD_VERSION_7.</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>SSLCipherSpec</strong></td>
<td>SSL Cipher Specification</td>
<td>650</td>
</tr>
<tr>
<td><strong>SSLPeerNamePtr</strong></td>
<td>SSL Peer Name Pointer</td>
<td>650</td>
</tr>
<tr>
<td><strong>SSLPeerNameLength</strong></td>
<td>SSL Peer Name Length</td>
<td>650</td>
</tr>
<tr>
<td><strong>SSLClientAuth</strong></td>
<td>SSL Client Authentication</td>
<td>651</td>
</tr>
<tr>
<td><strong>KeepAliveInterval</strong></td>
<td>Keepalive interval</td>
<td>651</td>
</tr>
</tbody>
</table>
**MQCD – Channel definition**

*Table 60. Fields in MQCD (continued)*

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>LocalAddress</td>
<td>Local communications address for the channel</td>
<td>651</td>
</tr>
<tr>
<td>BatchHeartbeat</td>
<td>Batch heartbeat interval</td>
<td>652</td>
</tr>
</tbody>
</table>

The MQCD structure contains the parameters which control execution of a channel. It is passed to each channel exit that is called from a Message Channel Agent (MCA). See MQ_CHANNEL_EXIT.

**Exit name fields**

When an exit is called, the relevant field from `SecurityExit, MsgExit, SendExit, ReceiveExit, and MsgRetryExit` contains the name of the exit currently being invoked. The meaning of the name in these fields depends on the environment in which the MCA is running. Except where noted below, the name is left-justified within the field, with no embedded blanks; the name is padded with blanks to the length of the field. In the descriptions that follow, square brackets ([ ]) denote optional information:

**UNIX systems**

The exit name is the name of a dynamically-loadable module or library, suffixed with the name of a function residing in that library. The function name must be enclosed in parentheses. The library name can optionally be prefixed with a directory path:

\[ \text{path} \text{library(function)} \]

The name is limited to a maximum of 128 characters.

**z/OS not using CICS for distributed queuing**

The exit name is the name of a load module that is valid for specification on the EP parameter of the LINK or LOAD macro. The name is limited to a maximum of 8 characters.

**z/OS using CICS for distributed queuing**

The exit name is a 4-character transaction identifier.

**OS/2, Windows, and DOS**

The exit name is the name of a dynamic-link library, suffixed with the name of a function residing in that library. The function name must be enclosed in parentheses. The library name can optionally be prefixed with a directory path and drive:

\[ \text{d:}\text{path}\text{library(function)} \]

The name is limited to a maximum of 128 characters.

**iSeries**

The exit name is a 10-byte program name followed by a 10-byte library name. If the names are less than 10 bytes long, each name is padded with blanks to make it 10 bytes. The library name can be *LIBL except when calling a channel auto-definition exit, in which case a fully qualified name is required.

**Fields**

**ChannelName (MQCHAR20)**

Channel definition name.
There must be a channel definition of the same name at the remote machine to be able to communicate.

The name must use only the characters:
- Uppercase A–Z
- Lowercase a–z
- Numerics 0–9
- Period (.)
- Forward slash (/)
- Underscore (_)
- Percent sign (%)

and be padded to the right with blanks. Leading or embedded blanks are not allowed.

The length of this field is given by MQ_CHANNEL_NAME_LENGTH.

**Version (MQLONG)**

Structure version number.

The value depends on the environment:

**MQCD_VERSION_1**

Version-1 channel definition structure.

The field has this value on WebSphere MQ for z/OS when CICS is used for distributed queuing. Note, however, that the MQCD passed to the exit is in fact a version-2 structure.

**MQCD_VERSION_2**

Version-2 channel definition structure.

This value is not used by any current WebSphere MQ product.

**MQCD_VERSION_3**

Version-3 channel definition structure.

The field has this value on MQSeries Version 2 in the following environments: Compaq OpenVMS Alpha, Compaq NonStop Kernel, UNIX systems not listed elsewhere, Windows systems.

**MQCD_VERSION_4**

Version-4 channel definition structure.

The field has this value on MQSeries Version 5 Release 0 in the following environments: AIX, HP-UX, OS/2, Solaris, Windows, and on MQSeries for MVS/ESA V1.2 when CICS is not used for distributed queuing.

**MQCD_VERSION_5**

Version-5 channel definition structure.

The field has this value on MQSeries for OS/390 Version 2 Release 1 and Version 5 Release 2 when CICS is not used for distributed queuing.

**MQCD_VERSION_6**

Version-6 channel definition structure.

The field has this value on MQSeries Version 5 Release 1 and Version 5 Release 2 in the following environments: AIX, HP-UX, OS/2, OS/400, Solaris, Windows, plus WebSphere MQ clients connected to these systems.

**MQCD_VERSION_7**

Version-7 channel definition structure.
The field has this value on Websphere MQ Version 5 Release 3 in the following environments: AIX, HP-UX, Solaris, Windows, and on Websphere MQ for z/OS Version 5 Release 3 when CICS is not used for distributed queuing.

Fields that exist only in the more-recent versions of the structure are identified as such in the descriptions of the fields. The following constant specifies the version number of the current version:

MQCD_CURRENT_VERSION
Current version of channel definition structure.

The value of this constant depends on the environment (see above).

Note: When a new version of the MQCD structure is introduced, the layout of the existing part is not changed. The exit should therefore check that the version number is equal to or greater than the lowest version which contains the fields that the exit needs to use.

**ChannelType (MQLONG)**
Channel type.

It is one of the following:

MQCHT_SENDER
Sender.

MQCHT_SERVER
Server.

MQCHT_RECEIVER
Receiver.

MQCHT_REQUESTER
Requester.

MQCHT_CLNTCONN
Client connection.

MQCHT_SVRCONN
Server-connection (for use by clients).

MQCHT_CLUSSDR
Cluster sender.

MQCHT_CLUSRCVR
Cluster receiver.

**TransportType (MQLONG)**
Transport type.

Transmission protocol to be used.

Note that the value will not have been checked if the channel was initiated from the other end.

The value is one of the following:

MQXPT_LU62
LU 6.2 transport protocol.

MQXPT_TCP
TCP/IP transport protocol.

MQXPT_NETBIOS
NetBIOS transport protocol.
MQCD – TransportType field

This value is supported in the following environments: OS/2, Windows.

MQXPT_SPX
SPX transport protocol.
This value is supported in the following environments: OS/2, Windows, plus WebSphere MQ clients connected to these systems.

MQXPT_DECNET
DECnet transport protocol.
This value is supported in the following environment: Compaq OpenVMS Alpha.

MQXPT_UDP
UDP transport protocol.
This value is supported in the following environments: AIX.

Desc (MQCHAR64)
Channel description.

This is a field that may be used for descriptive commentary. The content of the field is of no significance to Message Channel Agents. However, it should contain only characters that can be displayed. It cannot contain any null characters; if necessary, it is padded to the right with blanks. In a DBCS installation, the field can contain DBCS characters (subject to a maximum field length of 64 bytes).

Note: If this field contains characters that are not in the queue manager’s character set (as defined by the CodedCharSetId queue manager attribute), those characters may be translated incorrectly if this field is sent to another queue manager.

The length of this field is given by MQ_CHANNEL_DESC_LENGTH.

QMgrName (MQCHAR48)
Queue-manager name.

For channels with a ChannelType other than MQCHT_CLNTCONN, this is the name of the queue manager that an exit can connect to, which on OS/2, UNIX systems, and Windows NT, is always nonblank.

The length of this field is given by MQ_Q_MGR_NAME_LENGTH.

XmitQName (MQCHAR48)
Transmission queue name.

The name of the transmission queue from which messages are retrieved.

This field is relevant only for channels with a ChannelType of MQCHT_SENDER or MQCHT_SERVER.

The length of this field is given by MQ_Q_NAME_LENGTH.

ShortConnectionName (MQCHAR20)
First 20 bytes of connection name.

If the Version field is MQCD_VERSION_1, ShortConnectionName contains the full connection name.
**MQCD – ShortConnectionName field**

If the `Version` field is MQCD_VERSION_2 or greater, `ShortConnectionName` contains the first 20 characters of the connection name. The full connection name is given by the `ConnectionName` field; `ShortConnectionName` and the first 20 characters of `ConnectionName` are identical.

See `ConnectionName` for details of the contents of this field.

**Note:** The name of this field was changed for MQCD_VERSION_2 and subsequent versions of MQCD; the field was previously called `ConnectionName`.

The length of this field is given by MQ_SHORT_CONN_NAME_LENGTH.

**MCAName (MQCHAR20)**

Reserved.

This is a reserved field; its value is blank.

The length of this field is given by MQ_MCA_NAME_LENGTH.

**ModeName (MQCHAR8)**

LU 6.2 Mode name.

This field is relevant only if the transmission protocol (TransportType) is MQXPT_LU62, and the `ChannelType` is not MQCHT_SVRCONN or MQCHT_RECEIVER.

On iSeries, z/OS without CICS, UNIX systems, and MQSeries for Windows, this field is always blank. The information is contained in the communications Side Object instead.

The length of this field is given by MQ_MODE_NAME_LENGTH.

**TpName (MQCHAR64)**

LU 6.2 transaction program name.

This field is relevant only if the transmission protocol (TransportType) is MQXPT_LU62, and the `ChannelType` is not MQCHT_SVRCONN or MQCHT_RECEIVER.

On OS/400, z/OS without CICS, UNIX systems, and MQSeries for Windows, this field is always blank. The information is contained in the communications Side Object instead.

The length of this field is given by MQ_TP_NAME_LENGTH.

**BatchSize (MQLONG)**

Batch size.

The maximum number of messages that can be sent through a channel before synchronizing the channel.

This field is not relevant for channels with a `ChannelType` of MQCHT_SVRCONN or MQCHT_CLNTCONN.

**DiscInterval (MQLONG)**

Disconnect interval.
MQCD – DiscInterval field

The maximum time in seconds for which the channel waits for a message to arrive on the transmission queue, before terminating the channel. A value of zero causes the MCA to wait indefinitely.

This field is relevant only for channels with a ChannelType of MQCHT_SENDER, MQCHT_SERVER, MQCHT_CLUSSDR, or MQCHT_CLUSRCVR.

**ShortRetryCount (MQLONG)**

Short retry count.

This is the maximum number of attempts that are made to connect to the remote machine, at intervals specified by ShortRetryInterval, before the (normally longer) LongRetryCount and LongRetryInterval are used.

This field is relevant only for channels with a ChannelType of MQCHT_REQUESTER (only for WebSphere MQ for z/OS using CICS distributed queuing), MQCHT_SENDER, MQCHT_SERVER, MQCHT_CLUSSDR, or MQCHT_CLUSRCVR.

**ShortRetryInterval (MQLONG)**

Short retry wait interval.

This is the maximum number of seconds to wait before reattempting connection to the remote machine. Note that the interval between retries may be extended if the channel has to wait to become active.

This field is relevant only for channels with a ChannelType of MQCHT_REQUESTER (only for WebSphere MQ for z/OS using CICS distributed queuing), MQCHT_SENDER, MQCHT_SERVER, MQCHT_CLUSSDR, or MQCHT_CLUSRCVR.

**LongRetryCount (MQLONG)**

Long retry count.

This count is used after the count specified by ShortRetryCount has been exhausted. It specifies the maximum number of further attempts that are made to connect to the remote machine, at intervals specified by LongRetryInterval, before logging an error to the operator.

This field is relevant only for channels with a ChannelType of MQCHT_REQUESTER (only for WebSphere MQ for z/OS using CICS distributed queuing), MQCHT_SENDER, MQCHT_SERVER, MQCHT_CLUSSDR, or MQCHT_CLUSRCVR.

**LongRetryInterval (MQLONG)**

Long retry wait interval.

This is the maximum number of seconds to wait before reattempting connection to the remote machine. Note that the interval between retries may be extended if the channel has to wait to become active.

This field is relevant only for channels with a ChannelType of MQCHT_REQUESTER (only for WebSphere MQ for z/OS using CICS distributed queuing), MQCHT_SENDER, MQCHT_SERVER, MQCHT_CLUSSDR, or MQCHT_CLUSRCVR.
MQCD – SecurityExit field

SecurityExit (MQCHARn)
Channel security exit name.

If this name is nonblank, the exit is called at the following times:
- Immediately after establishing a channel.
  Before any messages are transferred, the exit is given the opportunity to
  instigate security flows to validate connection authorization.
- Upon receipt of a response to a security message flow.
  Any security message flows received from the remote processor on the remote
  machine are given to the exit.
- At initialization and termination of the channel.

See “Exit name fields” on page 628 for a description of the content of this field in
various environments.

The length of this field is given by MQ_EXIT_NAME_LENGTH.

Note: The value of this constant is environment specific.

MsgExit (MQCHARn)
Channel message exit name.

If this name is nonblank, the exit is called at the following times:
- Immediately after a message has been retrieved from the transmission queue
  (sender or server), or immediately before a message is put to a destination
  queue (receiver or requester).
  The exit is given the entire application message and transmission queue header
  for modification.
- At initialization and termination of the channel.

This field is not relevant for channels with a ChannelType of MQCHT_SVRCONN
or MQCHT_CLNTCONN; a message exit is never invoked for such channels.

See “Exit name fields” on page 628 for a description of the content of this field in
various environments.

The length of this field is given by MQ_EXIT_NAME_LENGTH.

Note: The value of this constant is environment specific.

SendExit (MQCHARn)
Channel send exit name.

If this name is nonblank, the exit is called at the following times:
- Immediately before data is sent out on the network.
  The exit is given the complete transmission buffer before it is transmitted. The
  contents of the buffer can be modified as required.
- At initialization and termination of the channel.

See “Exit name fields” on page 628 for a description of the content of this field in
various environments.

The length of this field is given by MQ_EXIT_NAME_LENGTH.
**ReceiveExit (MQCHARn)**

Channel receive exit name.

If this name is nonblank, the exit is called at the following times:

- Immediately before the received network data is processed.
  
  The exit is given the complete transmission buffer as received. The contents of the buffer can be modified as required.
- At initialization and termination of the channel.

See [“Exit name fields” on page 628](#) for a description of the content of this field in various environments.

The length of this field is given by MQ_EXIT_NAME_LENGTH.

**Note:** The value of this constant is environment specific.

**SeqNumberWrap (MQLONG)**

Highest allowable message sequence number.

When this value is reached, sequence numbers wrap to start again at 1.

This value is non-negotiable and must match in both the local and remote channel definitions.

This field is not relevant for channels with a `ChannelType` of MQCHT_SVRCONN or MQCHT_CLNTCONN.

**MaxMsgLength (MQLONG)**

Maximum message length.

Specifies the maximum message length that can be transmitted on the channel. This is compared with the value for the remote channel and the actual maximum is the lower of the two values.

**PutAuthority (MQLONG)**

Put authority.

Specifies whether the user identifier in the context information associated with a message should be used to establish authority to put the message to the destination queue.

This field is relevant only for channels with a `ChannelType` of MQCHT_REQUESTER, MQCHT_RECEIVER, or MQCHT_CLUSRCVR. It is one of the following:

- **MQPA_DEFAULT**
  
  Default user identifier is used.

- **MQPA_CONTEXT**
  
  Context user identifier is used.

**DataConversion (MQLONG)**

Data conversion.
MQCD – DataConversion field

This specifies whether the sending message channel agent should attempt conversion of the application message data if the receiving message channel agent is unable to perform this conversion. This applies only to messages that are not segments of logical messages; the MCA never attempts to convert messages which are segments.

This field is relevant only for channels with a ChannelType of MQCHT_SENDER, MQCHT_SERVER, MQCHT_CLUSSDR, or MQCHT_CLUSRCVR. It is one of the following:

MQCDC_SENDER_CONVERSION
  Conversion by sender.

MQCDC_NO_SENDER_CONVERSION
  No conversion by sender.

SecurityUserData (MQCHAR32)
Channel security exit user data.

This is passed to the channel security exit in the ExitData field of the ChannelExitParms parameter (see MQ_CHANNEL_EXIT).

This field initially contains the data that was set in the channel definition. However, during the lifetime of this MCA instance, any changes made to the contents of this field by an exit of any type are preserved by the MCA, and made visible to subsequent invocations of exits (regardless of type) for this MCA instance. Such changes have no effect on the channel definition used by other MCA instances. Any characters (including binary data) can be used.

The length of this field is given by MQ_EXIT_DATA_LENGTH.

MsgUserData (MQCHAR32)
Channel message exit user data.

This is passed to the channel message exit in the ExitData field of the ChannelExitParms parameter (see MQ_CHANNEL_EXIT).

This field initially contains the data that was set in the channel definition. However, during the lifetime of this MCA instance, any changes made to the contents of this field by an exit of any type are preserved by the MCA, and made visible to subsequent invocations of exits (regardless of type) for this MCA instance. Such changes have no effect on the channel definition used by other MCA instances. Any characters (including binary data) can be used.

The length of this field is given by MQ_EXIT_DATA_LENGTH.

SendUserData (MQCHAR32)
Channel send exit user data.

This is passed to the channel send exit in the ExitData field of the ChannelExitParms parameter (see MQ_CHANNEL_EXIT).

This field initially contains the data that was set in the channel definition. However, during the lifetime of this MCA instance, any changes made to the contents of this field by an exit of any type are preserved by the MCA, and made visible to subsequent invocations of exits (regardless of type) for this MCA instance. Such changes have no effect on the channel definition used by other MCA instances. Any characters (including binary data) can be used.
The length of this field is given by MQ_EXIT_DATA_LENGTH.

**ReceiveUserData (MQCHAR32)**
Channel receive exit user data.

This is passed to the channel receive exit in the ExitData field of the ChannelExitParms parameter (see MQ_CHANNEL_EXIT).

This field initially contains the data that was set in the channel definition. However, during the lifetime of this MCA instance, any changes made to the contents of this field by an exit of any type are preserved by the MCA, and made visible to subsequent invocations of exits (regardless of type) for this MCA instance. Such changes have no effect on the channel definition used by other MCA instances. Any characters (including binary data) can be used.

The length of this field is given by MQ_EXIT_DATA_LENGTH.

The following fields in this structure are not present if Version is less than MQCD_VERSION_2.

**UserIdentifier (MQCHAR12)**
User identifier.

This is used by the message channel agent when attempting to initiate a secure SNA session with a remote message channel agent.

This field can be nonblank only on OS/2, UNIX systems, and Windows, and is relevant only for channels with a ChannelType of MQCHT_SENDER, MQCHT_SERVER, MQCHT_REQUESTER or MQCHT_CLNTCONN. On z/OS this field is not relevant.

The length of this field is given by MQ_USER_ID_LENGTH, however only the first 10 characters are used.

This field is not present when Version is less than MQCD_VERSION_2.

**Password (MQCHAR12)**
Password.

This is used by the message channel agent when attempting to initiate a secure SNA session with a remote message channel agent.

This field can be nonblank only on OS/2, UNIX systems, and Windows, and is relevant only for channels with a ChannelType of MQCHT_SENDER, MQCHT_SERVER, MQCHT_REQUESTER or MQCHT_CLNTCONN. On z/OS this field is not relevant.

The length of this field is given by MQ_PASSWORD_LENGTH, however only the first 10 characters are used.

This field is not present if Version is less than MQCD_VERSION_2.

**MCAUserIdentifier (MQCHAR12)**
First 12 bytes of MCA user identifier.

There are two fields that contain the MCA user identifier:
MQCD – MCAUserIdentifier field

- **MCAUserIdentifier** contains the first 12 bytes of the MCA user identifier, and is padded with blanks if the identifier is shorter than 12 bytes. **MCAUserIdentifier** can be completely blank.
- **LongMCAUserIdPtr** points to the full MCA user identifier, which can be longer than 12 bytes. Its length is given by **LongMCAUserIdLength**. The full identifier contains no trailing blanks, and is not null-terminated. If the identifier is completely blank, **LongMCAUserIdLength** is zero, and the value of **LongMCAUserIdPtr** is undefined.

**Note:** **LongMCAUserIdPtr** is not present if **Version** is less than **MQCD_VERSION_6**.

If the MCA user identifier is nonblank, it specifies the user identifier to be used by the message channel agent for authorization to access WebSphere MQ resources, including (if **PutAuthority** is MQPA_DEFAULT) authorization to put the message to the destination queue for receiver or requester channels.

If the MCA user identifier is blank, the message channel agent uses its default user identifier.

The MCA user identifier can be set by a security exit to indicate the user identifier that the message channel agent should use. The exit can change either **MCAUserIdentifier**, or the string pointed at by **LongMCAUserIdPtr**. If both are changed but differ from each other, the MCA uses **LongMCAUserIdPtr** in preference to **MCAUserIdentifier**. If the exit changes the length of the string addressed by **LongMCAUserIdPtr**, **LongMCAUserIdLength** must be set correspondingly. If the exit wishes to increase the length of the identifier, the exit must allocate storage of the required length, set that storage to the required identifier, and place the address of that storage in **LongMCAUserIdPtr**. The exit is responsible for freeing that storage when the exit is later invoked with the MQXRTERM reason.

For channels with a **ChannelType** of MQCHT_SVRCONN, if **MCAUserIdentifier** in the channel definition is blank, any user identifier transferred from the client is copied into it. This user identifier (after any modification by the security exit at the server) is the one which the client application is assumed to be running under.

The MCA user identifier is not relevant for channels with a **ChannelType** of MQCHT_CLNTCONN.

This is an input/output field to the exit. The length of this field is given by **MQ_USER_ID_LENGTH**. This field is not present when **Version** is less than **MQCD_VERSION_2**.

**MCAType (MQLONG)**

Message channel agent type.

This is the type of the message channel agent program.

This field is relevant only for channels with a **ChannelType** of MQCHT_SENDER, MQCHT_SERVER, MQCHT_REQUESTER, MQCHT_CLUSSDR, or MQCHT_CLUSRCVR.

The value is one of the following:

**MQM_CAT_PROCESS**

Process.
The message channel agent runs as a separate process.

**MQMCAT_THREAD**

Thread (OS/2, iSeries, UNIX, and Windows).

The message channel agent runs as a separate thread.

This value is supported in the following environments: OS/2, Windows.

This field is not present when Version is less than MQCD_VERSION_2.

**ConnectionName (MQCHAR264)**

Connection name.

This is the full connection name of the partner. The type of name depends on the transmission protocol (TransportType) to be used:

- For MQXPT_LU62, it is the fully-qualified name of the partner Logical Unit.
- For MQXPT_NETBIOS, it is the NetBIOS name defined on the remote machine.
- For MQXPT_TCP, it is either the host name, or the network address of the remote machine.
- For MQXPT_SPX, it is an SPX-style address comprising a 4-byte network address, a 6-byte node address, and a 2-byte socket number.

When defining a channel, this field is not relevant for channels with a ChannelType of MQCHT_SVRCONN or MQCHT_RECEIVER. However, when the channel definition is passed to an exit, this field contains the address of the partner, whatever the channel type.

The length of this field is given by MQ_CONN_NAME_LENGTH. This field is not present if Version is less than MQCD_VERSION_2.

**RemoteUserIdentifier (MQCHAR12)**

First 12 bytes of user identifier from partner.

There are two fields that contain the remote user identifier:

- **RemoteUserIdentifier** contains the first 12 bytes of the remote user identifier, and is padded with blanks if the identifier is shorter than 12 bytes. RemoteUserIdentifier can be completely blank.

- **LongRemoteUserIdPtr** points to the full remote user identifier, which can be longer than 12 bytes. Its length is given by LongRemoteUserIdLength. The full identifier contains no trailing blanks, and is not null-terminated. If the identifier is completely blank, LongRemoteUserIdLength is zero, and the value of LongRemoteUserIdPtr is undefined.

  LongRemoteUserIdPtr is not present if Version is less than MQCD_VERSION_6.

The remote user identifier is relevant only for channels with a ChannelType of MQCHT_CLNTCONN or MQCHT_SVRCONN.

- For a security exit on an MQCHT_CLNTCONN channel, this is a user identifier that has been obtained from the environment (from an environment variable on OS/2 or from the system on UNIX platforms and Windows.) The exit can choose to send it to the security exit at the server.

- For a security exit on an MQCHT_SVRCONN channel, this field may contain a user identifier which has been obtained from the environment at the client, if there is no client security exit. The exit may validate this user ID (possibly in conjunction with the password in RemotePassword) and update the value in MCAUserIdentifier.
MQCD – RemoteUserIdentifier field

If there is a security exit at the client, then this information can be obtained in a security flow from the client.

The length of this field is given by MQ_USER_ID_LENGTH. This field is not present if Version is less than MQCD_VERSION_2.

RemotePassword (MQCHAR12)
Password from partner.

This field contains valid information only if ChannelType is MQCHT_CLNTCONN or MQCHT_SVRCONN.

- For a security exit at an MQCHT_CLNTCONN channel, this is a password which has been obtained from the environment from an environment variable on OS/2 and Windows. The exit can choose to send it to the security exit at the server.
- For a security exit at an MQCHT_SVRCONN channel, this field may contain a password which has been obtained from the environment at the client, if there is no client security exit. The exit may use this to validate the user identifier in RemoteUserIdentifier.

If there is a security exit at the client, then this information can be obtained in a security flow from the client.

The length of this field is given by MQ_PASSWORD_LENGTH. This field is not present if Version is less than MQCD_VERSION_2.

The following fields in this structure are not present if Version is less than MQCD_VERSION_3.

MsgRetryExit (MQCHARn)
Channel message retry exit name.

The message retry exit is an exit that is invoked by the MCA when the MCA receives a completion code of MQCC_FAILED from an MQOPEN or MQPUT call. The purpose of the exit is to specify a time interval for which the MCA should wait before retrying the MQOPEN or MQPUT operation. Alternatively, the exit can decide that the operation should not be retried.

The exit is invoked for all reason codes that have a completion code of MQCC_FAILED — it is up to the exit to decide which reason codes it wants the MCA to retry, for how many attempts, and at what time intervals.

When the exit decides that the operation should not be retried any more, the MCA performs its normal failure processing; this includes generating an exception report message (if specified by the sender), and either placing the original message on the dead-letter queue or discarding the message (according to whether the sender specified MQRO_DEAD_LETTER_Q or MQRO_DISCARD_MSG, respectively). Note that failures involving the dead-letter queue (for example, dead-letter queue full) do not cause the message-retry exit to be invoked.

If the exit name is nonblank, the exit is called at the following times:
- Immediately before performing the wait prior to retrying a message
- At initialization and termination of the channel.

See "Exit name fields” on page 628 for a description of the content of this field in various environments.
This field is relevant only for channels with a ChannelType of MQCHT_REQUESTER, MQCHT_RECEIVER, or MQCHT_CLUSRCVR.

The length of this field is given by MQ_EXIT_NAME_LENGTH.

Notes:
1. The value of this constant is environment specific.
2. On z/OS this field is not relevant.

This field is not present when Version is less than MQCD_VERSION_3.

**MsgRetryUserData (MQCHAR32)**
Channel message retry exit user data.

This is passed to the channel message-retry exit in the ExitData field of the ChannelExitParms parameter (see MQ_CHANNEL_EXIT).

This field initially contains the data that was set in the channel definition. However, during the lifetime of this MCA instance, any changes made to the contents of this field by an exit of any type are preserved by the MCA, and made visible to subsequent invocations of exits (regardless of type) for this MCA instance. Such changes have no effect on the channel definition used by other MCA instances. Any characters (including binary data) can be used.

This field is relevant only for channels with a ChannelType of MQCHT_REQUESTER, MQCHT_RECEIVER, or MQCHT_CLUSRCVR.

The length of this field is given by MQ_EXIT_DATA_LENGTH. This field is not present when Version is less than MQCD_VERSION_3.

On z/OS this field is always blank.

**MsgRetryCount (MQLONG)**
Number of times MCA will try to put the message, after the first attempt has failed.

This indicates the number of times that the MCA will retry the open or put operation, if the first MQOPEN or MQPUT fails with completion code MQCC_FAILED. The effect of this attribute depends on whether MsgRetryExit is blank or nonblank:

- **If MsgRetryExit is blank**, the MsgRetryCount attribute controls whether the MCA attempts retries. If the attribute value is zero, no retries are attempted. If the attribute value is greater than zero, the retries are attempted at intervals given by the MsgRetryInterval attribute.
  
  Retries are attempted only for the following reason codes:
  
  MQRC_PAGESET_FULL
  MQRC_PUT_INHIBITED
  MQRC_Q_FULL

  For other reason codes, the MCA proceeds immediately to its normal failure processing, without retrying the failing message.

- **If MsgRetryExit is nonblank**, the MsgRetryCount attribute has no effect on the MCA; instead it is the message-retry exit which determines how many times the retry is attempted, and at what intervals; the exit is invoked even if the MsgRetryCount attribute is zero.
MQCD – MsgRetryCount field

The `MsgRetryCount` attribute is made available to the exit in the MQCD structure, but the exit is not required to honor it — retries continue indefinitely until the exit returns `MQXCC_SUPPRESS_FUNCTION` in the `ExitResponse` field of MQCXP.

This field is relevant only for channels with a `ChannelType` of MQCHT_REQUESTER, MQCHT_RECEIVER, or MQCHT_CLUSRCVR.

This field is not present when `Version` is less than MQCD_VERSION_3.

On z/OS this field is always zero.

**MsgRetryInterval (MQLONG)**
Minimum interval in milliseconds after which the open or put operation will be retried.

The effect of this attribute depends on whether `MsgRetryExit` is blank or nonblank:

- If `MsgRetryExit` is blank, the `MsgRetryInterval` attribute specifies the minimum period of time that the MCA will wait before retrying a message, if the first MQOPEN or MQPUT fails with completion code MQCC_FAILED. A value of zero means that the retry will be performed as soon as possible after the previous attempt. Retries are performed only if `MsgRetryCount` is greater than zero.

  This attribute is also used as the wait time if the message-retry exit returns an invalid value in the `MsgRetryInterval` field in MQCXP.

- If `MsgRetryExit` is not blank, the `MsgRetryInterval` attribute has no effect on the MCA; instead it is the message-retry exit which determines how long the MCA should wait. The `MsgRetryInterval` attribute is made available to the exit in the MQCD structure, but the exit is not required to honor it.

  The interpretation of this field depends on the channel type, as follows:

  - For a channel type of MQCHT_SENDER, MQCHT_SERVER, MQCHT_RECEIVER, MQCHT_CLUSRCVR, or MQCHT_CLUSSDR, this is the time in seconds between heartbeat flows passed from the sending MCA when there are no messages on the transmission queue. This gives the receiving MCA the opportunity to quiesce the channel. To be useful, `HeartbeatInterval` should be significantly less than `DiscInterval`.

This type of heartbeat is supported in the following environments: AIX, HP-UX, Linux, z/OS, OS/2, iSeries, Solaris, Windows.
For a channel type of MQCHT_CLNTCONN or MQCHT_SVRCONN, this is the time in seconds between heartbeat flows passed from the server MCA when that MCA has issued an MQGET call with the MQGMO_WAIT option on behalf of a client application. This allows the server MCA to handle situations where the client connection fails during an MQGET with MQGMO_WAIT.

This type of heartbeat is supported in the following environments: AIX, HP-UX, Linux, OS/2, iSeries, Solaris, Windows.

The value is in the range 0 through 999 999. A value of 0 means that no heartbeat exchange occurs. The value that is actually used is the larger of the values specified at the sending side and receiving side.

This is an input field to the exit. The field is not present if Version is less than MQCD_VERSION_4.

**BatchInterval (MQLONG)**

Batch duration.

This is the approximate time in milliseconds that a channel will keep a batch open, if fewer than BatchSize messages have been transmitted in the current batch.

If BatchInterval is greater than zero, the batch is terminated by whichever of the following occurs first:
- BatchSize messages have been sent, or
- BatchInterval milliseconds have elapsed since the start of the batch.

If BatchInterval is zero, the batch is terminated by whichever of the following occurs first:
- BatchSize messages have been sent, or
- the transmission queue becomes empty.

BatchInterval must be in the range zero through 999 999 999.

This field is relevant only for channels with a ChannelType of MQCHT_SENDER, MQCHT_SERVER, MQCHT_CLUSSDR, or MQCHT_CLUSRCVR.

This is an input field to the exit. The field is not present when Version is less than MQCD_VERSION_4.

**NonPersistentMsgSpeed (MQLONG)**

Speed at which nonpersistent messages are sent.

This specifies the speed at which nonpersistent messages travel through the channel.

This field is relevant only for channels with a ChannelType of MQCHT_SENDER, MQCHT_SERVER, MQCHT_RECEIVER, MQCHT_REQUESTER, MQCHT_CLUSSDR, or MQCHT_CLUSRCVR.

The value is one of the following:

**MQNPMS_NORMAL**

Normal speed.

If a channel is defined to be MQNPMS_NORMAL, nonpersistent messages travel through the channel at normal speed. This has the advantage that
MQCD – NonPersistentMsgSpeed field

These messages will not be lost if there is a channel failure. Also, persistent and nonpersistent messages on the same transmission queue maintain their order relative to each other.

MQNPMFS_FAST
Fast speed.

If a channel is defined to be MQNPMFS_FAST, nonpersistent messages travel through the channel at fast speed. This improves the throughput of the channel, but means that nonpersistent messages will be lost if there is a channel failure. Also, it is possible for nonpersistent messages to jump ahead of persistent messages waiting on the same transmission queue, that is, the order of nonpersistent messages is not maintained relative to persistent messages. However the order of nonpersistent messages relative to each other is maintained. Similarly, the order of persistent messages relative to each other is maintained.

StrucLength (MQLONG)
Length of MQCD structure.

This is the length in bytes of the MQCD structure. The length does not include any of the strings addressed by pointer fields contained within the structure. The value is one of the following:

MQCD_LENGTH_4
Length of version-4 channel definition structure.

MQCD_LENGTH_5
Length of version-5 channel definition structure.

MQCD_LENGTH_6
Length of version-6 channel definition structure.

MQCD_LENGTH_7
Length of version-7 channel definition structure.

The following constant specifies the length of the current version:

MQCD_CURRENT_LENGTH
Length of current version of channel definition structure.

Note: These constants have values that are environment specific.

The field is not present if Version is less than MQCD_VERSION_4.

ExitNameLength (MQLONG)
Length of exit name.

This is the length in bytes of each of the names in the lists of exit names addressed by the MsgExitPtr, SendExitPtr, and ReceiveExitPtr fields. This length is not necessarily the same as MQ_EXIT_NAME_LENGTH.

This is an input field to the exit. The field is not present if Version is less than MQCD_VERSION_4.

ExitDataLength (MQLONG)
Length of exit user data.
This is the length in bytes of each of the user data items in the lists of exit user data items addressed by the `MsgUserDataPtr`, `SendUserDataPtr`, and `ReceiveUserDataPtr` fields. This length is not necessarily the same as `MQ_EXIT_DATA_LENGTH`.

This is an input field to the exit. The field is not present if `Version` is less than `MQCD_VERSION_4`.

**MsgExitsDefined (MQLONG)**

Number of message exits defined.

This is the number of channel message exits in the chain. It is greater than or equal to zero.

This is an input field to the exit. The field is not present if `Version` is less than `MQCD_VERSION_4`.

**SendExitsDefined (MQLONG)**

Number of send exits defined.

This is the number of channel send exits in the chain. It is greater than or equal to zero.

This is an input field to the exit. The field is not present if `Version` is less than `MQCD_VERSION_4`.

**ReceiveExitsDefined (MQLONG)**

Number of receive exits defined.

This is the number of channel receive exits in the chain. It is greater than or equal to zero.

This is an input field to the exit. The field is not present if `Version` is less than `MQCD_VERSION_4`.

**MsgExitPtr (MQPTR)**

Address of first `MsgExit` field.

If `MsgExitsDefined` is greater than zero, this is the address of the list of names of each channel message exit in the chain.

Each name is in a field of length `ExitNameLength`, padded to the right with blanks. There are `MsgExitsDefined` fields adjoining one another – one for each exit.

Any changes made to these names by an exit are preserved, although the message channel exit takes no explicit action – it does not change which exits are invoked.

If `MsgExitsDefined` is zero, this field is the null pointer.

On platforms where the programming language does not support the pointer data type, this field is declared as a byte string of the appropriate length.

This is an input field to the exit. The field is not present if `Version` is less than `MQCD_VERSION_4`.

**MsgUserDataPtr (MQPTR)**

Address of first `MsgUserData` field.
MQCD – MsgUserDataPtr field

If MsgExitsDefined is greater than zero, this is the address of the list of user data items for each channel message exit in the chain.

Each user data item is in a field of length ExitDataLength, padded to the right with blanks. There are MsgExitsDefined fields adjoining one another – one for each exit. If the number of user data items defined is less than the number of exit names, undefined user data items are set to blanks. Conversely, if the number of user data items defined is greater than the number of exit names, the excess user data items are ignored and not presented to the exit.

Any changes made to these values by an exit are preserved. This allows one exit to pass information to another exit. No validation is carried out on any changes so, for example, binary data can be written to these fields if required.

If MsgExitsDefined is zero, this field is the null pointer.

On platforms where the programming language does not support the pointer data type, this field is declared as a byte string of the appropriate length.

This is an input field to the exit. The field is not present if Version is less than MQCD_VERSION_4.

SendExitPtr (MQPTR)
Address of first SendExit field.

If SendExitsDefined is greater than zero, this is the address of the list of names of each channel send exit in the chain.

Each name is in a field of length ExitNameLength, padded to the right with blanks. There are SendExitsDefined fields adjoining one another – one for each exit.

Any changes made to these names by an exit are preserved, although the message send exit takes no explicit action – it does not change which exits are invoked.

If SendExitsDefined is zero, this field is the null pointer.

On platforms where the programming language does not support the pointer data type, this field is declared as a byte string of the appropriate length.

This is an input field to the exit. The field is not present if Version is less than MQCD_VERSION_4.

SendUserDataPtr (MQPTR)
Address of first SendUserData field.

If SendExitsDefined is greater than zero, this is the address of the list of user data items for each channel message exit in the chain.

Each user data item is in a field of length ExitDataLength, padded to the right with blanks. There are MsgExitsDefined fields adjoining one another – one for each exit. If the number of user data items defined is less than the number of exit names, undefined user data items are set to blanks. Conversely, if the number of user data items defined is greater than the number of exit names, the excess user data items are ignored and not presented to the exit.
Any changes made to these values by an exit are preserved. This allows one exit to pass information to another exit. No validation is carried out on any changes so, for example, binary data can be written to these fields if required.

If SendExitsDefined is zero, this field is the null pointer.

On platforms where the programming language does not support the pointer data type, this field is declared as a byte string of the appropriate length.

This is an input field to the exit. The field is not present if Version is less than MQCD_VERSION_4.

ReceiveExitPtr (MQPTR)
Address of first ReceiveExit field.

If ReceiveExitsDefined is greater than zero, this is the address of the list of names of each channel receive exit in the chain.

Each name is in a field of length ExitNameLength, padded to the right with blanks. There are ReceiveExitsDefined fields adjoining one another – one for each exit.

Any changes made to these names by an exit are preserved, although the message channel exit takes no explicit action – it does not change which exits are invoked.

If ReceiveExitsDefined is zero, this field is the null pointer.

On platforms where the programming language does not support the pointer data type, this field is declared as a byte string of the appropriate length.

This is an input field to the exit. The field is not present if Version is less than MQCD_VERSION_4.

ReceiveUserDataPtr (MQPTR)
Address of first ReceiveUserData field.

If ReceiveExitsDefined is greater than zero, this is the address of the list of user data item for each channel receive exit in the chain.

Each user data item is in a field of length ExitDataLength, padded to the right with blanks. There are ReceiveExitsDefined fields adjoining one another – one for each exit. If the number of user data items defined is less than the number of exit names, undefined user data items are set to blanks. Conversely, if the number of user data items defined is greater than the number of exit names, the excess user data items are ignored and not presented to the exit.

Any changes made to these values by an exit are preserved. This allows one exit to pass information to another exit. No validation is carried out on any changes so, for example, binary data can be written to these fields if required.

If ReceiveExitsDefined is zero, this field is the null pointer.

On platforms where the programming language does not support the pointer data type, this field is declared as a byte string of the appropriate length.

This is an input field to the exit. The field is not present if Version is less than MQCD_VERSION_4.
MQCD – ReceiveUserDataPtr field

The following fields in this structure are not present if Version is less than MQCD_VERSION_5.

**ClusterPtr (MQPTR)**
Address of a list of cluster names.

If ClustersDefined is greater than zero, this is the address of a list of cluster names. The channel belongs to each cluster listed.

This field is relevant only for channels with a ChannelType of MQCHT_CLUSSDR or MQCHT_CLUSRCVR.

This is an input field to the exit. The field is not present if Version is less than MQCD_VERSION_5.

**ClustersDefined (MQLONG)**
Number of clusters to which the channel belongs.

This is the number of cluster names pointed to by ClusterPtr. It is zero or greater.

This field is relevant only for channels with a ChannelType of MQCHT_CLUSSDR or MQCHT_CLUSRCVR.

This is an input field to the exit. The field is not present if Version is less than MQCD_VERSION_5.

**NetworkPriority (MQLONG)**
Network priority.

This is the priority of the network connection for this channel. When multiple paths to a particular destination are available, the path with the highest priority is chosen. The value is in the range 0 through 9; 0 is the lowest priority.

This field is relevant only for channels with a ChannelType of MQCHT_CLUSSDR or MQCHT_CLUSRCVR.

This is an input field to the exit. The field is not present if Version is less than MQCD_VERSION_5.

The following fields in this structure are not present if Version is less than MQCD_VERSION_6.

**LongMCAUserIdLength (MQLONG)**
Length of long MCA user identifier.

This is the length in bytes of the full MCA user identifier pointed to by LongMCAUserIdPtr.

This field is not relevant for channels with a ChannelType of MQCHT_CLNTCONN.

This is an input/output field to the exit. The field is not present if Version is less than MQCD_VERSION_6.

**LongRemoteUserIdLength (MQLONG)**
Length of long remote user identifier.
This is the length in bytes of the full remote user identifier pointed to by `LongRemoteUserIdPtr`.

This field is relevant only for channels with a `ChannelType` of MQCHT_CLNTCONN or MQCHT_SVRCONN.

This is an input field to the exit. The field is not present if `Version` is less than MQCD_VERSION_6.

**LongMCAUserIdPtr (MQPTR)**
Address of long MCA user identifier.

If `LongMCAUserIdLength` is greater than zero, this is the address of the full MCA user identifier. The length of the full identifier is given by `LongMCAUserIdLength`. The first 12 bytes of the MCA user identifier are also contained in the field `MCAUserIdentifier`.

See the description of the `MCAUserIdentifier` field for details of the MCA user identifier.

This field is not relevant for channels with a `ChannelType` of MQCHT_CLNTCONN.

This is an input/output field to the exit. The field is not present if `Version` is less than MQCD_VERSION_6.

**LongRemoteUserIdPtr (MQPTR)**
Address of long remote user identifier.

If `LongRemoteUserIdLength` is greater than zero, this is the address of the full remote user identifier. The length of the full identifier is given by `LongRemoteUserIdLength`. The first 12 bytes of the remote user identifier are also contained in the field `RemoteUserIdentifier`.

See the description of the `RemoteUserIdentifier` field for details of the remote user identifier.

This field is relevant only for channels with a `ChannelType` of MQCHT_CLNTCONN or MQCHT_SVRCONN.

This is an input field to the exit. The field is not present if `Version` is less than MQCD_VERSION_6.

**MCASecurityId (MQBYTE40)**
MCA security identifier.

This is the security identifier for the MCA.

This field is not relevant for channels with a `ChannelType` of MQCHT_CLNTCONN.

The following special value indicates that there is no security identifier:

**MQSID_NONE**
No security identifier specified.

The value is binary zero for the length of the field.
MQCD – MCASecurityId field

For the C programming language, the constant MQSID_NONE_ARRAY is also defined; this has the same value as MQSID_NONE, but is an array of characters instead of a string.

This is an input/output field to the exit. The length of this field is given by MQ_SECURITY_ID_LENGTH. This field is not present if Version is less than MQCD_VERSION_6.

RemoteSecurityId (MQBYTE40)
Remote security identifier.

This is the security identifier for the remote user.

This field is relevant only for channels with a ChannelType of MQCHT_CLNTCONN or MQCHT_SVRCONN.

The following special value indicates that there is no security identifier:

MQSID_NONE
No security identifier specified.

The value is binary zero for the length of the field.

For the C programming language, the constant MQSID_NONE_ARRAY is also defined; this has the same value as MQSID_NONE, but is an array of characters instead of a string.

This is an input field to the exit. The length of this field is given by MQ_SECURITY_ID_LENGTH. This field is not present if Version is less than MQCD_VERSION_6.

The following fields in this structure are not present if Version is less than MQCD_VERSION_7.

SSLCipherSpec (MQCHAR32)
SSL CipherSpec is an optional field.

This parameter is valid for all channel types. It is supported on AIX, HP-UX, Linux, iSeries, Solaris, Windows, and z/OS. It is valid only for channel types of a transport type (TRPTYPE) of TCP.

This is an input field to the exit. The length of this field is given by MQ_SSL_CIPHER_SPEC_LENGTH. The field is not present if Version is less than MQCD_VERSION_7.

SSLPeerNamePtr (MQPTR)
Address of the SSL peer name.

This is an input field to the exit. The field is not present if Version is less than MQCD_VERSION_7.

SSLPeerNameLength (MQLONG)
Length of SSL peer name.

This is the length in bytes of SSL peer name pointed to by SSLPeerNamePtr.

This is an input field to the exit. The field is not present if Version is less than MQCD_VERSION_7.
SSLClientAuth (MQLONG)
Determines whether SSL client authentication is required.

The value is one of the following:

MQSCA_REQUIRED
Client authentication required.

MQSCA_OPTIONAL
Client authentication optional.

This is an input field to the exit. The field is not present if Version is less than MQCD_VERSION_7.

KeepAliveInterval (MQLONG)
Keepalive interval.

This is the value passed to the communications stack for keepalive timing for the channel. The value is applicable for the TCP/IP and SPX communications protocols, though not all implementations support this parameter.

The value is in the range 0 through 99 999; the units are seconds. A value of zero indicates that channel keepalive is not enabled, although keepalive may still occur if TCP/IP keepalive (rather than channel keepalive) is enabled. The following special value is also valid:

MQKAI_AUTO
Automatic.

This indicates that the keepalive interval is calculated from the negotiated heartbeat interval, as follows:

- If the negotiated heartbeat interval is greater than zero, the keepalive interval that is used is the heartbeat interval plus 60 seconds.
- If the negotiated heartbeat interval is zero, the keepalive interval that is used is zero.

- On z/OS, TCP/IP keepalive occurs when TCPKEEP=YES is specified in CSQ6CHIP.
- In other environments, TCP/IP keepalive occurs when the KEEPALIVE=YES parameter is specified in the TCP stanza in the distributed queuing configuration file.

This field is relevant only for channels that have a TransportType of MQXPT_TCP or MQXPT_SPX.

This is an input field to the exit. The field is not present if Version is less than MQCD_VERSION_7.

LocalAddress (MQCHAR48)
Local communications address.

This is the local TCP/IP address defined for the channel for outbound communications, or blank if no specific address is defined for outbound communications. The address can optionally include a port number or range of port numbers. The format of this address is:

[ip-addr][[low-port[, high-port]]]
MQCD - LocalAddress field

where square brackets ([ ]) denote optional information, ip-addr is specified in
dotted decimal or alphanumeric form, and low-port and high-port are port
numbers enclosed in parentheses. All are optional.

A specific IP address, port, or port range for outbound communications is useful in
recovery scenarios where a channel is restarted on a different TCP/IP stack.

LocalAddress is similar in form to ConnectionName, but should not be confused
with it. LocalAddress specifies the characteristics of the local communications,
whereas ConnectionName specifies how to reach a remote queue manager.

This field is relevant only for channels with a TransportType of MQXPT_TCP, and
a ChannelType of MQCHT_SENDER, MQCHT_SERVER, MQCHT_REQUESTER,
MQCHT_CLNTCONN, MQCHT_CLUSSDR, or MQCHT_CLUSRCVR.

The length of this field is given by MQ_LOCAL_ADDRESS_LENGTH. This field is
not present if Version is less than MQCD_VERSION_7.

BatchHeartbeat (MQLONG)

Batch heartbeat interval.

This specifies the time interval that is used to trigger a batch heartbeat for the
channel. Batch hearbeating allows sender channels to determine whether the
remote channel instance is still active before going indoubt. A batch heartbeat
occurs if a sender channel has not communicated with the remote channel instance
within the specified time interval.

The value is in the range 0 through 999 999; the units are milliseconds. A value of
zero indicates that batch heartbeating is not enabled.

This field is relevant only for channels that have a ChannelType of
MQCHT_SENDER, MQCHT_SERVER, MQCHT_CLUSSDR, or
MQCHT_CLUSRCVR.

This is an input field to the exit. The field is not present if Version is less than
MQCD_VERSION_7.

C declaration

typedef struct tagMQCD MQCD;
struct tagMQCD {
  MQCHAR ChannelName[20]; /* Channel definition name */
  MQLONG Version; /* Structure version number */
  MQLONG ChannelType; /* Channel type */
  MQLONG TransportType; /* Transport type */
  MQCHAR Desc[64]; /* Channel description */
  MQCHAR QMgrName[48]; /* Queue-manager name */
  MQCHAR XmitQName[48]; /* Transmission queue name */
  MQCHAR ShortConnectionName[20]; /* First 20 bytes of connection
  MQCHAR MCAName[20]; /* Reserved */
  MQCHAR ModeName[8]; /* LU 6.2 Mode name */
  MQCHAR TpName[64]; /* LU 6.2 transaction program
  MQLONG BatchSize; /* Batch size */
  MQLONG DiscInterval; /* Disconnect interval */
  MQLONG ShortRetryCount; /* Short retry count */
  MQLONG ShortRetryInterval; /* Short retry wait interval */
  MQLONG LongRetryCount; /* Long retry count */
  MQLONG LongRetryInterval; /* Long retry wait interval */
}
MQCHAR SecurityExit[n]; /* Channel security exit name */
MQCHAR MsgExit[n]; /* Channel message exit name */
MQCHAR SendExit[n]; /* Channel send exit name */
MQCHAR ReceiveExit[n]; /* Channel receive exit name */
MQLONG SeqNumberWrap; /* Highest allowable message sequence number */
MQLONG MaxMsgLength; /* Maximum message length */
MQLONG PutAuthority; /* Put authority */
MQLONG DataConversion; /* Data conversion */
MQCHAR SecurityUserData[32]; /* Channel security exit user data */
MQCHAR MsgUserData[32]; /* Channel message exit user data */
MQCHAR SendUserData[32]; /* Channel send exit user data */
MQCHAR ReceiveUserData[32]; /* Channel receive exit user data */
MQCHAR UserIdentifier[12]; /* User identifier */
MQCHAR Password[12]; /* Password */
MQCHAR MCAUserIdentifier[12]; /* First 12 bytes of MCA user identifier */
MQLONG MCAType; /* Message channel agent type */
MQCHAR ConnectionName[264]; /* Connection name */
MQCHAR RemoteUserIdentifier[12]; /* First 12 bytes of user identifier from partner */
MQCHAR RemotePassword[12]; /* Password from partner */
MQCHAR MsgRetryExit[n]; /* Channel message retry exit name */
MQCHAR MsgRetryUserData[32]; /* Channel message retry exit user data */
MQCHAR MsgRetryCount; /* Number of times MCA will try to put the message, after the first attempt has failed */
MQLONG MsgRetryInterval; /* Minimum interval in milliseconds after which the open or put operation will be retried */
MQLONG HeartbeatInterval; /* Time in seconds between heartbeat flows */
MQLONG BatchInterval; /* Batch duration */
MQLONG NonPersistentMsgSpeed; /* Speed at which nonpersistent messages are sent */
MQLONG StrucLength; /* Length of MQCD structure */
MQLONG ExitNameLength; /* Length of exit name */
MQLONG ExitDataLength; /* Length of exit user data */
MQLONG MsgExitsDefined; /* Number of message exits defined */
MQLONG SendExitsDefined; /* Number of send exits defined */
MQLONG ReceiveExitsDefined; /* Number of receive exits defined */
MQPTR MsgExitPtr; /* Address of first MsgExit field */
MQPTR MsgUserDataPtr; /* Address of first MsgUserData field */
MQPTR SendExitPtr; /* Address of first SendExit field */
MQPTR SendUserDataPtr; /* Address of first SendUserData field */
MQPTR ReceiveExitPtr; /* Address of first ReceiveExit field */
MQPTR ReceiveUserDataPtr; /* Address of first ReceiveUserData field */
MQPTR ClusterPtr; /* Address of a list of cluster names */
MQLONG ClustersDefined; /* Number of clusters to which the channel belongs */
MQLONG NetworkPriority; /* Network priority */
MQCD – Language declarations

```c
MQLONG LongMCAUserIdLength; /* Length of long MCA user identifier */
MQLONG LongRemoteUserIdLength; /* Length of long remote user identifier */
MQPTR LongMCAUserIdPtr; /* Address of long MCA user identifier */
MQPTR LongRemoteUserIdPtr; /* Address of long remote user identifier */
MQBYTE40 MCASecurityId; /* MCA security identifier */
MQBYTE40 RemoteSecurityId; /* Remote security identifier */
MQPTR SSLPeerNamePtr; /* Address of SSL peer name */
MQLONG SSLPeerNameLength; /* Length of SSL peer name */
MQLONG SSLClientAuth; /* Whether SSL client authentication is required */
MQLONG KeepAliveInterval; /* Keepalive interval */
MQCHAR LocalAddress[48]; /* Local communications address */
MQLONG BatchHeartbeat; /* Batch heartbeat interval */
```

COBOL declaration

```cobol
** MQCD structure
10 MQCD.
** Channel definition name
15 MQCD-CHANNELNAME PIC X(20).
** Structure version number
15 MQCD-VERSION PIC S9(9) BINARY.
** Channel type
15 MQCD-CHANNELTYPE PIC S9(9) BINARY.
** Transport type
15 MQCD-TRANSPORTTYPE PIC S9(9) BINARY.
** Channel description
15 MQCD-DESC PIC X(64).
** Queue-manager name
15 MQCD-QMGRNAME PIC X(48).
** Transmission queue name
15 MQCD-XMITQNAME PIC X(48).
** First 20 bytes of connection name
15 MQCD-SHORTCONNECTIONNAME PIC X(20).
** Reserved
15 MQCD-MCANAME PIC X(20).
** LU 6.2 Mode name
15 MQCD-MODENAME PIC X(8).
** LU 6.2 transaction program name
15 MQCD-TPNAME PIC X(64).
** Batch size
15 MQCD-BATCHSIZE PIC S9(9) BINARY.
** Disconnect interval
15 MQCD-DISCINTERVAL PIC S9(9) BINARY.
** Short retry count
15 MQCD-SHORTRETRYCOUNT PIC S9(9) BINARY.
** Short retry wait interval
15 MQCD-SHORTRETRYINTERVAL PIC S9(9) BINARY.
** Long retry count
15 MQCD-LONGRETRYCOUNT PIC S9(9) BINARY.
** Long retry wait interval
15 MQCD-LONGRETRYINTERVAL PIC S9(9) BINARY.
** Channel security exit name
15 MQCD-SECURITYEXIT PIC X(n).
** Channel message exit name
15 MQCD-MEGEXIT PIC X(n).
** Channel send exit name
15 MQCD-SENDEXIT PIC X(n).
** Channel receive exit name
15 MQCD-RECEIVEEXIT PIC X(n).
```
** Highest allowable message sequence number
15 MQCD-SEQNUMBERWRAP PIC S9(9) BINARY.
** Maximum message length
15 MQCD-MAXMSGLENGTH PIC S9(9) BINARY.
** Put authority
15 MQCD-PUTAUTHORITY PIC S9(9) BINARY.
** Data conversion
15 MQCD-DATACONVERSION PIC S9(9) BINARY.
** Channel security exit user data
  15 MQCD-SECURITYUSERDATA PIC X(32).
** Channel message exit user data
  15 MQCD-MSGUSERDATA PIC X(32).
** Channel send exit user data
  15 MQCD-SENDUSERDATA PIC X(32).
** Channel receive exit user data
  15 MQCD-RECEIVEUSERDATA PIC X(32).
** User identifier
  15 MQCD-USERIDENTIFIER PIC X(12).
** Password
  15 MQCD-PASSWORD PIC X(12).
** First 12 bytes of MCA user identifier
  15 MQCD-MCAUSERIDENTIFIER PIC X(12).
** Message channel agent type
  15 MQCD-MCATYPE PIC S9(9) BINARY.
** Connection name
  15 MQCD-CONNECTIONNAME PIC X(264).
** First 12 bytes of user identifier from partner
  15 MQCD-REMOTEUSERIDENTIFIER PIC X(12).
** Password from partner
  15 MQCD-REMOTEPASSWORD PIC X(12).
** Channel message retry exit name
  15 MQCD-MSGRETRYEXIT PIC X(n).
** Channel message retry exit user data
  15 MQCD-MSGRETRYUSERDATA PIC X(32).
** Number of times MCA will try to put the message, after the first attempt has failed
  15 MQCD-MSGRETRYCOUNT PIC S9(9) BINARY.
** Minimum interval in milliseconds after which the open or put operation will be retried
  15 MQCD-MSGRETRYINTERVAL PIC S9(9) BINARY.
** Time in seconds between heartbeat flows
  15 MQCD-HEARTBEATINTERVAL PIC S9(9) BINARY.
** Batch duration
  15 MQCD-BATCHINTERVAL PIC S9(9) BINARY.
** Speed at which nonpersistent messages are sent
  15 MQCD-NONPERSISTENTMSGSPEED PIC S9(9) BINARY.
** Length of MQCD structure
  15 MQCD-STRUCLENGTH PIC S9(9) BINARY.
** Length of exit name
  15 MQCD-EXITNAMELENGTH PIC S9(9) BINARY.
** Length of exit user data
  15 MQCD-EXITDATALENGTH PIC S9(9) BINARY.
** Number of message exits defined
  15 MQCD-MSGEXITSDEFINED PIC S9(9) BINARY.
** Number of send exits defined
  15 MQCD-SENDEXITSDEFINED PIC S9(9) BINARY.
** Number of receive exits defined
  15 MQCD-RECEIVEEXITSDEFINED PIC S9(9) BINARY.
** Address of first MSGEXIT field
  15 MQCD-MSGEXITPTR POINTER.
** Address of first MSGUSERDATA field
  15 MQCD-MSGUSERDATAPTR POINTER.
** Address of first SENDEXIT field
  15 MQCD-SENDEXITPTR POINTER.
** Address of first SENDUSERDATA field
  15 MQCD-SENDUSERDATAPTR POINTER.
** Address of first RECEIVEEXIT field
MQCD – Language declarations

15 MQCD-RECEIVEEXITPTR POINTER.
** Address of first RECEIVEUSERDATA field
15 MQCD-RECEIVEUSERDATAPTR POINTER.
** Address of a list of cluster names
15 MQCD-CLUSTERPTR POINTER.
** Number of clusters to which the channel belongs
15 MQCD-CLUSTERSDEFINED PIC S9(9) BINARY.
** Network priority
15 MQCD-NETWORKPRIORITY PIC S9(9) BINARY.
** Length of long MCA user identifier
15 MQCD-LONGMAUSERIDLENGTH PIC S9(9) BINARY.
** Length of long remote user identifier
15 MQCD-LONGREMOTEUSERIDLENGTH PIC S9(9) BINARY.
** Address of long MCA user identifier
15 MQCD-LONGMAUSERIDPTR POINTER.
** Address of long remote user identifier
15 MQCD-LONGREMOTEUSERIDPTR POINTER.
** MCA security identifier
15 MQCD-MCASECURITYID PIC X(40).
** Remote security identifier
15 MQCD-REMOTESECURITYID PIC X(40).
** SSL CipherSpec
15 MQCD-SSLCSPEC PIC X(32).
** Address of SSL peer name
15 MQCD-SSLPEERNAMEPTR POINTER.
** Length of SSL peer name
15 MQCD-SSLPEERNAMELENGTH PIC S9(9) BINARY.
** Whether SSL client authentication is required
15 MQCD-SSLCLIENTAUTH PIC S9(9) BINARY.
** Keepalive interval
15 MQCD-KEEPALIVEINTERVAL PIC S9(9) BINARY.
** Local communications address
15 MQCD-LOCALADDRESS PIC X(48).
** Batch heartbeat interval
15 MQCD-BATCHHEARTBEAT PIC S9(9) BINARY.

PL/I declaration
dcl
1 MQCD based,
  3 ChannelName char(20), /* Channel definition name */
  3 Version fixed bin(31), /* Structure version number */
  3 ChannelType fixed bin(31), /* Channel type */
  3 TransportType fixed bin(31), /* Transport type */
  3 Desc char(64), /* Channel description */
  3 QMgrName char(48), /* Queue-manager name */
  3 XmitQName char(48), /* Transmission queue name */
  3 ShortConnectionName char(20), /* First 20 bytes of
collection name */
  3 MCAName char(20), /* Reserved */
  3 ModeName char(8), /* LU 6.2 Mode name */
  3 TpName char(64), /* LU 6.2 transaction program name */
  3 BatchSize fixed bin(31), /* Batch size */
  3 DiscInterval fixed bin(31), /* Disconnect interval */
  3 ShortRetryCount fixed bin(31), /* Short retry count */
  3 ShortRetryInterval fixed bin(31), /* Short retry wait
interval */
  3 LongRetryCount fixed bin(31), /* Long retry count */
  3 LongRetryInterval fixed bin(31), /* Long retry wait interval */
  3 SecurityExit char(n), /* Channel security exit name */
  3 MsgExit char(n), /* Channel message exit name */
  3 SendExit char(n), /* Channel send exit name */
  3 ReceiveExit char(n), /* Channel receive exit name */
MQCD – Language declarations

3 SeqNumberWrap fixed bin(31), /* Highest allowable message sequence number */
3 MaxMsgLength fixed bin(31), /* Maximum message length */
3 PutAuthority fixed bin(31), /* Put authority */
3 DataConversion fixed bin(31), /* Data conversion */
3 SecurityUserData char(32), /* Channel security exit user data */
3 MsgUserData char(32), /* Channel message exit user data */
3 SendUserData char(32), /* Channel send exit user data */
3 ReceiveUserData char(32), /* Channel receive exit user data */
3 UserIdentifier char(12), /* User identifier */
3 Password char(12), /* Password */
3 MCAUserIdentifier char(12), /* First 12 bytes of MCA user identifier */
3 MCAType fixed bin(31), /* Message channel agent type */
3 ConnectionName char(264), /* Connection name */
3 RemoteUserIdentifier char(12), /* First 12 bytes of user identifier from partner */
3 RemotePassword char(12), /* Password from partner */
3 MsgRetryExit char(n), /* Channel message retry exit name */
3 MsgRetryUserData char(32), /* Channel message retry exit user data */
3 MsgRetryCount fixed bin(31), /* Number of times MCA will try to put the message, after the first attempt has failed */
3 MsgRetryInterval fixed bin(31), /* Minimum interval in milliseconds after which the open or put operation will be retried */
3 HeartbeatInterval fixed bin(31), /* Time in seconds between heartbeat flows */
3 BatchInterval fixed bin(31), /* Batch duration */
3 NonPersistentMsgSpeed fixed bin(31), /* Speed at which nonpersistent messages are sent */
3 StrucLength fixed bin(31), /* Length of MQCD structure */
3 ExitNameLength fixed bin(31), /* Length of exit name */
3 ExitDataLength fixed bin(31), /* Length of exit user data */
3 MsgExitsDefined fixed bin(31), /* Number of message exits defined */
3 SendExitsDefined fixed bin(31), /* Number of send exits defined */
3 ReceiveExitsDefined fixed bin(31), /* Number of receive exits defined */
3 MsgExitPtr pointer, /* Address of first MsgExit field */
3 MsgUserDataPtr pointer, /* Address of first MsgUserData field */
3 SendExitPtr pointer, /* Address of first SendExit field */
3 SendUserDataPtr pointer, /* Address of first SendUserData field */
3 ReceiveExitPtr pointer, /* Address of first ReceiveExit field */
3 ReceiveUserDataPtr pointer, /* Address of first ReceiveUserData field */
3 ClusterPtr pointer, /* Address of a list of cluster names */
3 ClustersDefined fixed bin(31), /* Number of clusters to which the channel belongs */
3 NetworkPriority fixed bin(31), /* Network priority */
MQCD – Language declarations

3 LongMCAUserIdLength fixed bin(31), /* Length of long MCA user identifier */
3 LongRemoteUserIdLength fixed bin(31), /* Length of long remote user identifier */
3 LongMCAUserIdPtr pointer, /* Address of long MCA user identifier */
3 LongRemoteUserIdPtr pointer, /* Address of long remote user identifier */
3 MCASecurityId char(40), /* MCA security identifier */
3 RemoteSecurityId char(40), /* Remote security identifier */
3 SSLCipherSpec char(32), /* SSL CipherSpec */
3 SSLPeerNamePtr pointer, /* Address of SSL peer name */
3 SSLPeerNameLength fixed bin(31), /* Length of SSL peer name */
3 SSLClientAuth fixed bin(31), /* Whether SSL client authentication is required */
3 KeepAliveInterval fixed bin(31), /* Keepalive interval */
3 LocalAddress char(48), /* Local communications address */
3 BatchHeartbeat fixed bin(31); /* Batch heartbeat interval */

RPG declaration (ILE)

D*..1....2....3....4....5....6....7...
D* MQCD Structure
D*
D* Channel definition name
D CDCHN 1 20
D* Structure version number
D COVER 21 24 I 0
D* Channel type
D CDCHT 25 28 I 0
D* Transport type
D CDTRT 29 32 I 0
D* Channel description
D CDES 33 96
D* Queue-manager name
D CDQM 97 144
D* Transmission queue name
D CDXQ 145 192
D* First 20 bytes of connection name
D CDSCN 193 212
D* Reserved
D CDMA 213 232
D* LU 6.2 Mode name
D CDMOD 233 240
D* LU 6.2 transaction program name
D CDTP 241 304
D* Batch size
D CBIS 305 308 I 0
D* Disconnect interval
D CDII 309 312 I 0
D* Short retry count
D CDSRC 313 316 I 0
D* Short retry wait interval
D CDSRI 317 320 I 0
D* Long retry count
D CDLRC 321 324 I 0
D* Long retry wait interval
D CDLRI 325 328 I 0
D* Channel security exit name
D CDSCX 329 348
D* Channel message exit name
D CDMSX 349 368
D* Channel send exit name
D CDSNX 369 388
<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
<th>Offset</th>
<th>Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>D* Channel receive exit name</td>
<td>CDRCX 389 408</td>
<td></td>
<td></td>
</tr>
<tr>
<td>D* Highest allowable message sequence number</td>
<td>CDSNW 409 412I 0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>D* Maximum message length</td>
<td>CDMMML 413 416I 0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>D* Put authority</td>
<td>CDPA 417 420I 0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>D* Data conversion</td>
<td>CDCC 421 424I 0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>D* Channel security exit user data</td>
<td>CDSCD 425 456</td>
<td></td>
<td></td>
</tr>
<tr>
<td>D* Channel message exit user data</td>
<td>CDMSD 457 488</td>
<td></td>
<td></td>
</tr>
<tr>
<td>D* Channel send exit user data</td>
<td>CDSND 489 520</td>
<td></td>
<td></td>
</tr>
<tr>
<td>D* Channel receive exit user data</td>
<td>CDRCRD 521 552</td>
<td></td>
<td></td>
</tr>
<tr>
<td>D* User identifier</td>
<td>CDUID 553 564</td>
<td></td>
<td></td>
</tr>
<tr>
<td>D* Password</td>
<td>CDPW 565 576</td>
<td></td>
<td></td>
</tr>
<tr>
<td>D* First 12 bytes of MCA user identifier</td>
<td>CDUID 577 588</td>
<td></td>
<td></td>
</tr>
<tr>
<td>D* Message channel agent type</td>
<td>CDCAT 589 592I 0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>D* Connection name (characters 1 through 256)</td>
<td>CDCON 593 848</td>
<td></td>
<td></td>
</tr>
<tr>
<td>D* Connection name (characters 257 through 264)</td>
<td>CDCN2 849 856</td>
<td></td>
<td></td>
</tr>
<tr>
<td>D* First 12 bytes of user identifier from partner</td>
<td>CDRUID 857 868</td>
<td></td>
<td></td>
</tr>
<tr>
<td>D* Password from partner</td>
<td>CDRPW 869 880</td>
<td></td>
<td></td>
</tr>
<tr>
<td>D* Channel message retry exit name</td>
<td>CDMRX 881 900</td>
<td></td>
<td></td>
</tr>
<tr>
<td>D* Channel message retry exit user data</td>
<td>CDMRD 901 932</td>
<td></td>
<td></td>
</tr>
<tr>
<td>D* Number of times MCA will try to put the message, after the first attempt has failed</td>
<td>CDMRC 933 936I 0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>D* Minimum interval in milliseconds after which the open or put operation will be retried</td>
<td>CDMRI 937 940I 0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>D* Time in seconds between heartbeat flows</td>
<td>CDHBI 941 944I 0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>D* Batch duration</td>
<td>CDBI 945 948I 0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>D* Speed at which nonpersistent messages are sent</td>
<td>CDNPM 949 952I 0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>D* Length of MQCD structure</td>
<td>CDLEN 953 956I 0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>D* Length of exit name</td>
<td>CDXNL 957 960I 0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>D* Length of exit user data</td>
<td>CDXDL 961 964I 0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>D* Number of message exits defined</td>
<td>CDMXD 965 968I 0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>D* Number of send exits defined</td>
<td>CDSXD 969 972I 0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>D* Number of receive exits defined</td>
<td>CDRXD 973 976I 0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>D* Address of first CDMSX field</td>
<td>CDMXP 977 992*</td>
<td></td>
<td></td>
</tr>
<tr>
<td>D* Address of first CDMSD field</td>
<td>CDMUP 993 1008*</td>
<td></td>
<td></td>
</tr>
<tr>
<td>D* Address of first CDSNX field</td>
<td>CDMUP 993 1008*</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
MQCD – Language declarations

D CDSXP 1009 1024*
D* Address of first CDSND field
D CDSUP 1025 1040*
D* Address of first CDRCX field
D CDRXP 1041 1056*
D* Address of first CDRCD field
D CDRUP 1057 1072*
D* Address of a list of cluster names
D CDCLP 1073 1088*
D* Number of clusters to which the channel belongs
D CDCLD 1089 1092 I 0
D* Network priority
D CDNP 1093 1096 I 0
D* Length of long MCA user identifier
D CDLML 1097 1100 I 0
D* Length of long remote user identifier
D CDLRL 1101 1104 I 0
D* Address of long MCA user identifier
D CDLMP 1105 1120 I 0
D* Address of long remote user identifier
D CDLRP 1121 1136 I 0
D* MCA security identifier
D CDMSI 1137 1176
D* Remote security identifier
D CDRSI 1177 1216
D* SSL CipherSpec
D CDSCS 1217 1248
D* Address of SSL peer name
D CDSPP 1249 1264 I 0
D* Length of SSL peer name
D CDSPL 1265 1268 I 0
D* Whether SSL client authentication is required
D CDSCA 1269 1272 I 0
D* Keepalive interval
D CDKAI 1273 1276 I 0
D* Local communications address
D CDLAD 1277 1324
D* Batch heartbeat interval
D CDBHB 1325 1328 I 0

RPG declaration (OPM)

I*.............2...........3...........4...........5...........6...........7...
I* MQCD Structure
I*
I* Channel definition name
I 1 20 CDCHN
I* Structure version number
I B 21 240CDVER
I* Channel type
I B 25 280CDCHT
I* Transport type
I B 29 320CDTRT
I* Channel description
I 33 96 CDDES
I* Queue-manager name
I 97 144 CDQM
I* Transmission queue name
I 145 192 CDXQ
I* First 20 bytes of connection name
I 193 212 CDSCN
I* Reserved
I 213 232 CDMCA
I* LU 6.2 Mode name
I 233 240 CDMOD
I* LU 6.2 transaction program name
I 241 304 CDTP
**MQCD – Language declarations**

<table>
<thead>
<tr>
<th>Variable Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Batch size</td>
<td>B 305 3080CDBS</td>
</tr>
<tr>
<td>Disconnect interval</td>
<td>B 309 3120CDDI</td>
</tr>
<tr>
<td>Short retry count</td>
<td>B 313 3160CDSRC</td>
</tr>
<tr>
<td>Short retry wait interval</td>
<td>B 317 3200CDSRI</td>
</tr>
<tr>
<td>Long retry count</td>
<td>B 321 3240CDLRC</td>
</tr>
<tr>
<td>Long retry wait interval</td>
<td>B 325 3280CDLRI</td>
</tr>
<tr>
<td>Channel security exit name</td>
<td>329 348 CDSCX</td>
</tr>
<tr>
<td>Channel message exit name</td>
<td>349 368 CDMSX</td>
</tr>
<tr>
<td>Channel send exit name</td>
<td>369 388 CDSNX</td>
</tr>
<tr>
<td>Channel receive exit name</td>
<td>389 408 CDRCX</td>
</tr>
<tr>
<td>Highest allowable message sequence number</td>
<td>B 409 4120CDSNW</td>
</tr>
<tr>
<td>Maximum message length</td>
<td>B 413 4160CDMML</td>
</tr>
<tr>
<td>Put authority</td>
<td>B 417 4200CDPA</td>
</tr>
<tr>
<td>Data conversion</td>
<td>B 421 4240CDDC</td>
</tr>
<tr>
<td>Channel security exit user data</td>
<td>425 456 CDSCD</td>
</tr>
<tr>
<td>Channel message exit user data</td>
<td>457 488 CDMSO</td>
</tr>
<tr>
<td>Channel send exit user data</td>
<td>489 520 CDSND</td>
</tr>
<tr>
<td>Channel receive exit user data</td>
<td>521 552 CDRCD</td>
</tr>
<tr>
<td>User identifier</td>
<td>553 564 CDUID</td>
</tr>
<tr>
<td>Password</td>
<td>565 576 CDPW</td>
</tr>
<tr>
<td>First 12 bytes of MCA user identifier</td>
<td>577 588 CDAUI</td>
</tr>
<tr>
<td>Message channel agent type</td>
<td>B 589 5920CDCAT</td>
</tr>
<tr>
<td>Connection name (characters 1 through 256)</td>
<td>593 848 CDCON</td>
</tr>
<tr>
<td>Connection name (characters 257 through 264)</td>
<td>849 856 CDCN2</td>
</tr>
<tr>
<td>First 12 bytes of user identifier from partner</td>
<td>857 868 CDRUI</td>
</tr>
<tr>
<td>Password from partner</td>
<td>869 880 CDRPW</td>
</tr>
<tr>
<td>Channel message retry exit name</td>
<td>881 900 CDMRX</td>
</tr>
<tr>
<td>Channel message retry exit user data</td>
<td>901 932 CDMRD</td>
</tr>
<tr>
<td>Number of times MCA will try to put the message, after the first attempt has failed</td>
<td>B 933 9360CDMRC</td>
</tr>
<tr>
<td>Minimum interval in milliseconds after which the open or put operation will be retried</td>
<td>B 937 9400CDMRI</td>
</tr>
<tr>
<td>Time in seconds between heartbeat flows</td>
<td>B 941 9440CDHBI</td>
</tr>
<tr>
<td>Batch duration</td>
<td>B 945 9480CDBI</td>
</tr>
<tr>
<td>Speed at which nonpersistent messages are sent</td>
<td></td>
</tr>
</tbody>
</table>
MQCD – Language declarations

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Length of MQCD structure</td>
<td>B 949</td>
</tr>
<tr>
<td>Length of exit name</td>
<td>B 953</td>
</tr>
<tr>
<td>Length of exit user data</td>
<td>B 957</td>
</tr>
<tr>
<td>Number of message exits defined</td>
<td>B 961</td>
</tr>
<tr>
<td>Number of send exits defined</td>
<td>B 965</td>
</tr>
<tr>
<td>Number of receive exits defined</td>
<td>B 969</td>
</tr>
<tr>
<td>Address of first CDMSX field</td>
<td>B 973</td>
</tr>
<tr>
<td>Address of first CDMSD field</td>
<td>B 977</td>
</tr>
<tr>
<td>Address of first CDSNX field</td>
<td>B 993</td>
</tr>
<tr>
<td>Address of first CDSND field</td>
<td>B 1009</td>
</tr>
<tr>
<td>Address of first CDRCX field</td>
<td>B 1025</td>
</tr>
<tr>
<td>Address of first CDRCD field</td>
<td>B 1041</td>
</tr>
<tr>
<td>Address of a list of cluster names</td>
<td>B 1073</td>
</tr>
<tr>
<td>Number of clusters to which the channel belongs</td>
<td>B 1089</td>
</tr>
<tr>
<td>Network priority</td>
<td>B 1093</td>
</tr>
<tr>
<td>Length of long MCA user identifier</td>
<td>B 1097</td>
</tr>
<tr>
<td>Length of long remote user identifier</td>
<td>B 1101</td>
</tr>
<tr>
<td>Address of long MCA user identifier</td>
<td>B 1105</td>
</tr>
<tr>
<td>Address of long remote user identifier</td>
<td>B 1121</td>
</tr>
<tr>
<td>MCA security identifier</td>
<td>B 1137</td>
</tr>
<tr>
<td>Remote security identifier</td>
<td>B 1177</td>
</tr>
<tr>
<td>SSL CipherSpec</td>
<td>B 1217</td>
</tr>
<tr>
<td>Address of SSL peer name</td>
<td>B 1249</td>
</tr>
<tr>
<td>Length of SSL peer name</td>
<td>B 1265</td>
</tr>
<tr>
<td>Whether SSL client authentication is required</td>
<td>B 1269</td>
</tr>
<tr>
<td>Keepalive interval</td>
<td>B 1273</td>
</tr>
<tr>
<td>Local communications address</td>
<td>B 1277</td>
</tr>
<tr>
<td>Batch heartbeat interval</td>
<td>B 1325</td>
</tr>
</tbody>
</table>

System/390 assembler declaration

<table>
<thead>
<tr>
<th>Field</th>
<th>DSECT</th>
</tr>
</thead>
<tbody>
<tr>
<td>MQCD</td>
<td>DSECT</td>
</tr>
<tr>
<td>MQCD_CHANNELNAME</td>
<td>DS CL20</td>
</tr>
<tr>
<td>MQCD_VERSION</td>
<td>DS F</td>
</tr>
<tr>
<td>MQCD_CHANNELTYPE</td>
<td>DS F</td>
</tr>
<tr>
<td>MQCD_TRANSPORTTYPE</td>
<td>DS F</td>
</tr>
<tr>
<td>MQCD_DESC</td>
<td>DS CL64</td>
</tr>
<tr>
<td>MQCD_QMGRNAME</td>
<td>DS CL48</td>
</tr>
<tr>
<td>Variable Name</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>MQCD_XMITQNAME</td>
<td>Transmission queue name</td>
</tr>
<tr>
<td>MQCD_SHORTCONNECTIONNAME</td>
<td>First 20 bytes of connection name</td>
</tr>
<tr>
<td>MQCD_MCNAME</td>
<td>Reserved</td>
</tr>
<tr>
<td>MQCD_MODENAME</td>
<td>LU 6.2 Mode name</td>
</tr>
<tr>
<td>MQCD_TPNAME</td>
<td>LU 6.2 transaction program name</td>
</tr>
<tr>
<td>MQCD_BATCHSIZE</td>
<td>Batch size</td>
</tr>
<tr>
<td>MQCD_DISCINTERVAL</td>
<td>Disconnect interval</td>
</tr>
<tr>
<td>MQCD_SHORTRETRYINTERVAL</td>
<td>Short retry wait interval</td>
</tr>
<tr>
<td>MQCD_LONGRETRYINTERVAL</td>
<td>Long retry count</td>
</tr>
<tr>
<td>MQCD_SECURITYEXIT</td>
<td>Channel security exit name</td>
</tr>
<tr>
<td>MQCD_MSGEXIT</td>
<td>Channel message exit name</td>
</tr>
<tr>
<td>MQCD_SENDEXIT</td>
<td>Channel send exit name</td>
</tr>
<tr>
<td>MQCD_RECEIVEEXIT</td>
<td>Channel receive exit name</td>
</tr>
<tr>
<td>MQCD_SEQNUMBERWRAP</td>
<td>Highest allowable message sequence number</td>
</tr>
<tr>
<td>MQCD_MAXMSGLENGTH</td>
<td>Maximum message length</td>
</tr>
<tr>
<td>MQCD_PUTAUTHORITY</td>
<td>Put authority</td>
</tr>
<tr>
<td>MQCD_DATACONVERSION</td>
<td>Data conversion</td>
</tr>
<tr>
<td>MQCD_SECURITYUSERDATA</td>
<td>Channel security exit user data</td>
</tr>
<tr>
<td>MQCD_MGUSERDATA</td>
<td>Channel message exit user data</td>
</tr>
<tr>
<td>MQCD_SENDUSERDATA</td>
<td>Channel send exit user data</td>
</tr>
<tr>
<td>MQCD_RECEIVEUSERDATA</td>
<td>Channel receive exit user data</td>
</tr>
<tr>
<td>MQCD_USERIDENTIFIER</td>
<td>User identifier</td>
</tr>
<tr>
<td>MQCD_PASSWORD</td>
<td>Password</td>
</tr>
<tr>
<td>MQCD_MCAUSERIDENTIFIER</td>
<td>First 12 bytes of MCA user identifier</td>
</tr>
<tr>
<td>MQCD_MCTYPE</td>
<td>Message channel agent type</td>
</tr>
<tr>
<td>MQCD_CONNECTIONNAME</td>
<td>Connection name</td>
</tr>
<tr>
<td>MQCD_REMOTEUSERIDENTIFIER</td>
<td>First 12 bytes of user identifier from partner</td>
</tr>
<tr>
<td>MQCD_REMOTEPASSWORD</td>
<td>Password from partner</td>
</tr>
<tr>
<td>MQCD_MSGRETRYEXIT</td>
<td>Channel message retry exit name</td>
</tr>
<tr>
<td>MQCD_MSGRETRYUSERDATA</td>
<td>Channel message retry exit user data</td>
</tr>
<tr>
<td>MQCD_MSGRETRYCOUNT</td>
<td>Number of times MCA will try to put the message, after the first attempt has failed</td>
</tr>
<tr>
<td>MQCD_MSGRETRYINTERVAL</td>
<td>Minimum interval in milliseconds after which the open or put operation will be retried</td>
</tr>
<tr>
<td>MQCD_HEARTBEATINTERVAL</td>
<td>Time in seconds between heartbeat flows</td>
</tr>
<tr>
<td>MQCD_BATCHINTERVAL</td>
<td>Batch duration</td>
</tr>
<tr>
<td>MQCD_NONPERSISTENTMSGSPEED</td>
<td>Speed at which nonpersistent messages are sent</td>
</tr>
<tr>
<td>MQCD_STRUCLength</td>
<td>Length of MQCD structure</td>
</tr>
<tr>
<td>MQCD_EXITNAMELENGTH</td>
<td>Length of exit name</td>
</tr>
<tr>
<td>MQCD_MGEXITSDefined</td>
<td>Length of exit user data</td>
</tr>
<tr>
<td>MQCD_SENDEXITSDefined</td>
<td>Number of message exits defined</td>
</tr>
<tr>
<td>MQCD_RECEIVEEXITSDefined</td>
<td>Number of receive exits defined</td>
</tr>
<tr>
<td>MQCD_MGEXITPTR</td>
<td>Address of first MSGEXIT field</td>
</tr>
<tr>
<td>MQCD_MGUSERDATAPTR</td>
<td>Address of first MSGUSERDATA field</td>
</tr>
<tr>
<td>MQCD_SENDEXITPTR</td>
<td>Address of first SENDEXIT field</td>
</tr>
<tr>
<td>MQCD_SENDUSERDATAPTR</td>
<td>Address of first SENDUSERDATA field</td>
</tr>
<tr>
<td>MQCD_RECEIVEEXITPTR</td>
<td>Address of first RECEIVEEXIT field</td>
</tr>
<tr>
<td>MQCD_RECEIVEUSERDATAPTR</td>
<td>Address of first RECEIVEUSERDATA field</td>
</tr>
<tr>
<td>MQCD_CLUSTERPTR</td>
<td>Address of a list of cluster names</td>
</tr>
</tbody>
</table>
### MQCD – Language declarations

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MQCD_CLUSTERSDEFINED</td>
<td>DS F</td>
<td>Number of clusters to which the channel belongs</td>
</tr>
<tr>
<td>MQCD_NETWORKPRIORITY</td>
<td>DS F</td>
<td>Network priority</td>
</tr>
<tr>
<td>MQCD_LONGMCAUSERIDLENGTH</td>
<td>DS F</td>
<td>Length of long MCA user identifier</td>
</tr>
<tr>
<td>MQCD_LONGREMOTEUSERIDLENGTH</td>
<td>DS F</td>
<td>Length of long remote user identifier</td>
</tr>
<tr>
<td>MQCD_LONGMCAUSERIDPTR</td>
<td>DS F</td>
<td>Address of long MCA user identifier</td>
</tr>
<tr>
<td>MQCD_LONGREMOTEUSERIDPTR</td>
<td>DS F</td>
<td>Address of long remote user identifier</td>
</tr>
<tr>
<td>MQCD_MCASECURITYID</td>
<td>DS XL40</td>
<td>MCA security identifier</td>
</tr>
<tr>
<td>MQCD_REMOTESECURITYID</td>
<td>DS XL40</td>
<td>Remote security identifier</td>
</tr>
<tr>
<td>MQCD_SSLCIPHERSPEC</td>
<td>DS CL32</td>
<td>SSL CipherSpec</td>
</tr>
<tr>
<td>MQCD_SSLPEERNAMEPTR</td>
<td>DS F</td>
<td>Address of SSL peer name</td>
</tr>
<tr>
<td>MQCD_SSLPEERNAMELENGTH</td>
<td>DS F</td>
<td>Length of SSL peer name</td>
</tr>
<tr>
<td>MQCD_SSLCLIENTAUTH</td>
<td>DS F</td>
<td>Whether SSL client authentication is required</td>
</tr>
<tr>
<td>MQCD_KEEPALIVEINTERVAL</td>
<td>DS F</td>
<td>Keepalive interval</td>
</tr>
<tr>
<td>MQCD_LOCALADDRESS</td>
<td>DS CL48</td>
<td>Local communications address</td>
</tr>
<tr>
<td>MQCD_BATCHHEARTBEAT</td>
<td>DS F</td>
<td>Batch heartbeat interval</td>
</tr>
<tr>
<td>MQCD_LENGTH</td>
<td>EQU</td>
<td>*-MQCD</td>
</tr>
<tr>
<td>MQCD_AREA</td>
<td>DS CL(MQCD_LENGTH)</td>
<td></td>
</tr>
</tbody>
</table>

### Visual Basic declaration

In Visual Basic, the MQCD structure can be used with the MQCNO structure on the MQCONNX call.

```vbnet
Type MQCD
    ChannelName As String*20 'Channel definition name'
    Version As Long 'Structure version number'
    ChannelType As Long 'Channel type'
    TransportType As Long 'Transport type'
    Desc As String*64 'Channel description'
    QMgrName As String*48 'Queue-manager name'
    XmitQName As String*48 'Transmission queue name'
    ShortConnectionName As String*20 'First 20 bytes of connection'
    MCAName As String*20 'Reserved'
    ModeName As String*8 'LU 6.2 Mode name'
    TpName As String*64 'LU 6.2 transaction program name'
    BatchSize As Long 'Batch size'
    DiscInterval As Long 'Disconnect interval'
    ShortRetryCount As Long 'Short retry count'
    ShortRetryInterval As Long 'Short retry wait interval'
    LongRetryCount As Long 'Long retry count'
    LongRetryInterval As Long 'Long retry wait interval'
    SecurityExit As String*n 'Channel security exit name'
    MsgExit As String*n 'Channel message exit name'
    SendExit As String*n 'Channel send exit name'
    ReceiveExit As String*n 'Channel receive exit name'
    SeqNumberWrap As Long 'Highest allowable message sequence number'
    MaxMsgLength As Long 'Maximum message length'
    PutAuthority As Long 'Put authority'
    DataConversion As Long 'Data conversion'
    SecurityUserData As String*32 'Channel security exit user data'
    MsgUserData As String*32 'Channel message exit user data'
    SendUserData As String*32 'Channel send exit user data'
    ReceiveUserData As String*32 'Channel receive exit user data'
    UserIdentifier As String*12 'User identifier'
    Password As String*12 'Password'
    MCAUserIdentifier As String*12 'First 12 bytes of MCA user identifier'
End Type
```
<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MCAType</td>
<td>As Long</td>
<td>'Message channel agent type'</td>
</tr>
<tr>
<td>ConnectionName</td>
<td>As String=264</td>
<td>'Connection name'</td>
</tr>
<tr>
<td>RemoteUserIdentifier</td>
<td>As String=12</td>
<td>'First 12 bytes of user identifier from partner'</td>
</tr>
<tr>
<td>RemotePassword</td>
<td>As String=12</td>
<td>'Password from partner'</td>
</tr>
<tr>
<td>MsgRetryExit</td>
<td>As String=n</td>
<td>'Channel message retry exit name'</td>
</tr>
<tr>
<td>MsgRetryUserData</td>
<td>As String=32</td>
<td>'Channel message retry exit user data'</td>
</tr>
<tr>
<td>MsgRetryCount</td>
<td>As Long</td>
<td>'Number of times MCA will try to put the message, after the first attempt has failed'</td>
</tr>
<tr>
<td>MsgRetryInterval</td>
<td>As Long</td>
<td>'Minimum interval in milliseconds after which the open or put operation will be retried'</td>
</tr>
<tr>
<td>HeartbeatInterval</td>
<td>As Long</td>
<td>'Time in seconds between heartbeat flows'</td>
</tr>
<tr>
<td>BatchInterval</td>
<td>As Long</td>
<td>'Batch duration'</td>
</tr>
<tr>
<td>NonPersistentMsgSpeed</td>
<td>As Long</td>
<td>'Speed at which nonpersistent messages are sent'</td>
</tr>
<tr>
<td>StrucLength</td>
<td>As Long</td>
<td>'Length of MQCD structure'</td>
</tr>
<tr>
<td>ExitNameLength</td>
<td>As Long</td>
<td>'Length of exit name'</td>
</tr>
<tr>
<td>ExitUserDataLength</td>
<td>As Long</td>
<td>'Length of exit user data'</td>
</tr>
<tr>
<td>MsgExitsDefined</td>
<td>As Long</td>
<td>'Number of message exits defined'</td>
</tr>
<tr>
<td>SendExitsDefined</td>
<td>As Long</td>
<td>'Number of send exits defined'</td>
</tr>
<tr>
<td>ReceiveExitsDefined</td>
<td>As Long</td>
<td>'Number of receive exits defined'</td>
</tr>
<tr>
<td>MsgExitPtr</td>
<td>As String=4</td>
<td>'Address of first MsgExit field'</td>
</tr>
<tr>
<td>MsgUserDataPtr</td>
<td>As String=4</td>
<td>'Address of first MsgUserData field'</td>
</tr>
<tr>
<td>SendExitPtr</td>
<td>As String=4</td>
<td>'Address of first SendExit field'</td>
</tr>
<tr>
<td>SendUserDataPtr</td>
<td>As String=4</td>
<td>'Address of first SendUserData field'</td>
</tr>
<tr>
<td>ReceiveExitPtr</td>
<td>As String=4</td>
<td>'Address of first ReceiveExit field'</td>
</tr>
<tr>
<td>ReceiveUserDataPtr</td>
<td>As String=4</td>
<td>'Address of first ReceiveUserData field'</td>
</tr>
<tr>
<td>ClusterPtr</td>
<td>As String=4</td>
<td>'Address of a list of cluster names'</td>
</tr>
<tr>
<td>ClustersDefined</td>
<td>As Long</td>
<td>'Number of clusters to which the channel belongs'</td>
</tr>
<tr>
<td>NetworkPriority</td>
<td>As Long</td>
<td>'Network priority'</td>
</tr>
<tr>
<td>LongMCAUserIdLength</td>
<td>As Long</td>
<td>'Length of long MCA user identifier'</td>
</tr>
<tr>
<td>LongRemoteUserIdLength</td>
<td>As Long</td>
<td>'Length of long remote user identifier'</td>
</tr>
<tr>
<td>LongMCAUserIdPtr</td>
<td>As String=4</td>
<td>'Address of long MCA user identifier'</td>
</tr>
<tr>
<td>LongRemoteUserIdPtr</td>
<td>As String=4</td>
<td>'Address of long remote user identifier'</td>
</tr>
<tr>
<td>MCASecurityId</td>
<td>As String=40</td>
<td>'MCA security identifier'</td>
</tr>
<tr>
<td>RemoteSecurityId</td>
<td>As String=40</td>
<td>'Remote security identifier'</td>
</tr>
<tr>
<td>SSLCipherSpec</td>
<td>As String=32</td>
<td>'SSL CipherSpec'</td>
</tr>
<tr>
<td>SSLPeerNamePtr</td>
<td>As String=4</td>
<td>'Address of SSL peer name'</td>
</tr>
<tr>
<td>SSLPeerNameLength</td>
<td>As Long</td>
<td>'Length of SSL peer name'</td>
</tr>
<tr>
<td>SSLClientAuth</td>
<td>As Long</td>
<td>'Whether SSL client'</td>
</tr>
<tr>
<td>KeepAliveInterval</td>
<td>As Long</td>
<td>'Keepalive interval'</td>
</tr>
<tr>
<td>LocalAddress</td>
<td>As String=48</td>
<td>'Local communications address'</td>
</tr>
<tr>
<td>BatchHeartbeat</td>
<td>As Long</td>
<td>'Batch heartbeat interval'</td>
</tr>
</tbody>
</table>
The following table summarizes the fields in the structure.

**Table 61. Fields in MQCXP**

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>StrucId</td>
<td>Structure identifier</td>
<td>666</td>
</tr>
<tr>
<td>Version</td>
<td>Structure version number</td>
<td>667</td>
</tr>
<tr>
<td>ExitId</td>
<td>Type of exit</td>
<td>667</td>
</tr>
<tr>
<td>ExitReason</td>
<td>Reason for invoking exit</td>
<td>668</td>
</tr>
<tr>
<td>ExitResponse</td>
<td>Response from exit</td>
<td>670</td>
</tr>
<tr>
<td>ExitResponse2</td>
<td>Secondary response from exit</td>
<td>671</td>
</tr>
<tr>
<td>Feedback</td>
<td>Feedback code</td>
<td>673</td>
</tr>
<tr>
<td>MaxSegmentLength</td>
<td>Maximum segment length</td>
<td>673</td>
</tr>
<tr>
<td>ExitUserArea</td>
<td>Exit user area</td>
<td>673</td>
</tr>
<tr>
<td>ExitData</td>
<td>Exit data</td>
<td>674</td>
</tr>
<tr>
<td>MsgRetryCount</td>
<td>Number of times the message has been retried</td>
<td>674</td>
</tr>
<tr>
<td>MsgRetryInterval</td>
<td>Minimum interval in milliseconds after which the put operation should be retried</td>
<td>674</td>
</tr>
<tr>
<td>MsgRetryReason</td>
<td>Reason code from previous attempt to put the message</td>
<td>674</td>
</tr>
<tr>
<td>HeaderLength</td>
<td>Length of header</td>
<td>675</td>
</tr>
<tr>
<td>PartnerName</td>
<td>Partner name</td>
<td>675</td>
</tr>
<tr>
<td>FAPLevel</td>
<td>Negotiated Formats and Protocols level</td>
<td>675</td>
</tr>
<tr>
<td>CapabilityFlags</td>
<td>Capability flags</td>
<td>675</td>
</tr>
<tr>
<td>ExitNumber</td>
<td>Exit number</td>
<td>676</td>
</tr>
<tr>
<td>ExitSpace</td>
<td>Number of bytes in transmission header reserved for exit to use</td>
<td>676</td>
</tr>
</tbody>
</table>

The MQCXP structure is passed to each type of exit called by a Message Channel Agent (MCA). See MQ_CHANNEL_EXIT.

The fields described as “input to the exit” in the descriptions that follow are ignored by the MCA when the exit returns control to the MCA. The exit should not expect that any input fields that it changes in the channel exit parameter block will be preserved for its next invocation. Changes made to input/output fields (for example, the ExitUserArea field), are preserved for invocations of that instance of the exit only. Such changes cannot be used to pass data between different exits defined on the same channel, or between the same exit defined on different channels.

**Fields**

**StrucId (MQCHAR4)**
Structure identifier.

The value must be:
MQCXP_STRUC_ID
Identifier for channel exit parameter structure.
For the C programming language, the constant
MQCXP_STRUC_ID_ARRAY is also defined; this has the same value as
MQCXP_STRUC_ID, but is an array of characters instead of a string.

This is an input field to the exit.

Version (MQLONG)
Structure version number.
The value depends on the environment:

MQCXP_VERSION_1
Version-1 channel exit parameter structure.
The field has this value on z/OS using CICS for distributed queuing.

MQCXP_VERSION_2
Version-2 channel exit parameter structure.
The field has this value in the following environments: Compaq OpenVMS Alpha, Compaq NonStop Kernel.

MQCXP_VERSION_3
Version-3 channel exit parameter structure.
The field has this value in the following environments: UNIX systems not listed elsewhere.

MQCXP_VERSION_4
Version-4 channel exit parameter structure.
The field has this value in the following environments: z/OS not using CICS for distributed queuing, OS/2.

MQCXP_VERSION_5
Version-5 channel exit parameter structure.
The field has this value in the following environments: AIX, HP-UX, Linux, iSeries, Solaris, Windows.

Fields that exist only in the more-recent versions of the structure are identified as such in the descriptions of the fields. The following constant specifies the version number of the current version:

MQCXP_CURRENT_VERSION
Current version of channel exit parameter structure.
The value of this constant depends on the environment (see above).

Note: When a new version of the MQCXP structure is introduced, the layout of the existing part is not changed. The exit should therefore check that the version number is equal to or greater than the lowest version which contains the fields that the exit needs to use.

This is an input field to the exit.

ExitId (MQLONG)
Type of exit.
MQCXP – ExitId field

This indicates the type of exit being called, and is set on entry to the exit routine. Possible values are:

MQXT_CHANNEL_SEC_EXIT
Channel security exit.

MQXT_CHANNEL_MSG_EXIT
Channel message exit.

MQXT_CHANNEL_SEND_EXIT
Channel send exit.

MQXT_CHANNEL_RCV_EXIT
Channel receive exit.

MQXT_CHANNEL_MSG_RETRY_EXIT
Channel message-retry exit.

This type of exit is not supported on: z/OS.

MQXT_CHANNEL_AUTO_DEF_EXIT
Channel auto-definition exit.

On z/OS, this type of exit is supported only for channels of type MQCHT_CLUSSDR and MQCHT_CLUSRCVR.

This is an input field to the exit.

ExitReason (MQLONG)
Reason for invoking exit.

This indicates the reason why the exit is being called, and is set on entry to the exit routine. It is not used by the auto-definition exit. Possible values are:

MQXR_INIT
Exit initialization.

This indicates that the exit is being invoked for the first time. It allows the exit to acquire and initialize any resources that it may need (for example: main storage).

MQXR_TERM
Exit termination.

This indicates that the exit is about to be terminated. The exit should free any resources that it may have acquired since it was initialized (for example: main storage).

MQXR_MSG
Process a message.

This indicates that the exit is being invoked to process a message. This occurs for channel message exits only.

MQXR_XMIT
Process a transmission.

This occurs for channel send and receive exits only.

MQXR_SEC_MSG
Security message received.

This occurs for channel security exits only.

MQXR_INIT_SEC
Initiate security exchange.
This occurs for channel security exits only.

The receiver’s security exit is always invoked with this reason immediately after being invoked with MQXR_INIT, to give it the opportunity to initiate a security exchange. If it declines the opportunity (by returning MQXCC_OK instead of MQXCC_SEND_SEC_MSG or MQXCC_SEND_AND_REQUEST_SEC_MSG), the sender’s security exit is invoked with MQXR_INIT_SEC.

If the receiver’s security exit does initiate a security exchange (by returning MQXCC_SEND_SEC_MSG or MQXCC_SEND_AND_REQUEST_SEC_MSG), the sender’s security exit is never invoked with MQXR_INIT_SEC; instead it is invoked with MQXR_SEC_MSG to process the receiver’s message. (In either case it is first invoked with MQXR_INIT.)

Unless one of the security exits requests termination of the channel (by setting ExitResponse to MQXCC_SUPPRESS_FUNCTION or MQXCC_CLOSE_CHANNEL), the security exchange must complete at the side that initiated the exchange. Therefore, if a security exit is invoked with MQXR_INIT_SEC and it does initiate an exchange, the next time the exit is invoked it will be with MQXR_SEC_MSG. This happens whether or not there is a security message for the exit to process. There will be a security message if the partner returns MQXCC_SEND_SEC_MSG or MQXCC_SEND_AND_REQUEST_SEC_MSG, but not if the partner returns MQXCC_OK or there is no security exit at the partner. If there is no security message to process, the security exit at the initiating end is re-invoked with a DataLength of zero.

MQXR_RETRY
Retry a message.
This occurs for message-retry exits only.
On z/OS, this is not supported.

MQXR_AUTO_CLUSSDR
Automatic definition of a cluster-sender channel.
This occurs for channel auto-definition exits only.

MQXR_AUTO_RECEIVER
Automatic definition of a receiver channel.
This occurs for channel auto-definition exits only.

MQXR_AUTO_SVRCONN
Automatic definition of a server-connection channel.
This occurs for channel auto-definition exits only.

MQXR_AUTO_CLUSRCVR
Automatic definition of a cluster-receiver channel.
This occurs for channel auto-definition exits only.

Notes:
1. If you have more than one exit defined for a channel, they will each be invoked with MQXR_INIT when the MCA is initialized, and will each be invoked with MQXR_TERM when the MCA is terminated.
2. For the channel auto-definition exit, ExitReason is not set if Version is less than MQCXP_VERSION_4. The value MQXR_AUTO_SVRCONN is implied in this case.
MQXP – ExitResponse field

This is an input field to the exit.

**ExitResponse (MQLONG)**

Response from exit.

This is set by the exit to communicate with the MCA. It must be one of the following:

**MQXCC_OK**

Exit completed successfully.

- For the channel security exit, this indicates that message transfer should now proceed normally.
- For the channel message retry exit, this indicates that the MCA should wait for the time interval returned by the exit in the **MsgRetryInterval** field in MQXP, and then retry the message.

The **ExitResponse2** field may contain additional information.

**MQXCC_SUPPRESS_FUNCTION**

Suppress function.

- For the channel security exit, this indicates that the channel should be terminated.
- For the channel message exit, this indicates that the message is not to proceed any further towards its destination. Instead the MCA generates an exception report message (if one was requested by the sender of the original message), and places the message contained in the original buffer on the dead-letter queue (if the sender specified MQRO_DEAD_LETTER_Q), or discards it (if the sender specified MQRO_DISCARD_MSG).

For persistent messages, if the sender specified MQRO_DEAD_LETTER_Q, but the put to the dead-letter queue fails, or there is no dead-letter queue, the original message is left on the transmission queue and the report message is not generated. The original message is also left on the transmission queue if the report message cannot be generated successfully.

The Feedback field in the MQDLH structure at the start of the message on the dead-letter queue indicates why the message was put on the dead-letter queue; this feedback code is also used in the message descriptor of the exception report message (if one was requested by the sender).

- For the channel message retry exit, this indicates that the MCA should not wait and retry the message; instead, the MCA continues immediately with its normal failure processing (the message is placed on the dead-letter queue or discarded, as specified by the sender of the message).
- For the channel auto-definition exit, either MQXCC_OK or MQXCC_SUPPRESS_FUNCTION must be specified. If neither of these is specified, MQXCC_SUPPRESS_FUNCTION is assumed by default and the auto-definition is abandoned.

This response is not supported for the channel send and receive exits.

**MQXCC_SEND_SEC_MSG**

Send security message.
MQXP – ExitResponse field

This value can be set only by a channel security exit. It indicates that the exit has provided a security message which should be transmitted to the partner.

MQXCC_SEND_AND_REQUEST_SEC_MSG
Send security message that requires a reply.
This value can be set only by a channel security exit. It indicates
• that the exit has provided a security message which should be transmitted to the partner, and
• that the exit requires a response from the partner. If no response is received, the channel must be terminated, because the exit has not yet decided whether communications can proceed.

This is not valid on z/OS if you are using CICS for distributed queuing.

MQXCC_SUPPRESS_EXIT
Suppress exit.
• This value can be set by all types of channel exit other than a security exit or an auto-definition exit. It suppresses any further invocation of that exit (as if its name had been blank in the channel definition), until termination of the MCA, when the exit is again invoked with an ExitReason of MQXR_TERM.
• If a message retry exit returns this value, message retries for subsequent messages are controlled by the MsgRetryCount and MsgRetryInterval channel attributes as normal. For the current message, the MCA performs the number of outstanding retries, at intervals given by the MsgRetryInterval channel attribute, but only if the reason code is one that the MCA would normally retry (see the MsgRetryCount field described in “MQCD – Channel definition” on page 626). The number of outstanding retries is the value of the MsgRetryCount attribute, less the number of times the exit returned MQXCC_OK for the current message; if this number is negative, no further retries are performed by the MCA for the current message.

This is not valid on z/OS if you are using CICS for distributed queuing.

MQXCC_CLOSE_CHANNEL
Close channel.
This value can be set by any type of channel exit except an auto-definition exit. It causes the message channel agent (MCA) to close the channel.

This is not valid on z/OS if you are using CICS for distributed queuing.

This is an input/output field from the exit.

ExitResponse2 (MQLONG)
Secondary response from exit.

This is set to zero on entry to the exit routine. It can be set by the exit to provide further information to the MCA. It is not used by the auto-definition exit.

The exit can set one or more of the following. If more than one is required, the values are added together. Combinations that are not valid are noted; other combinations are allowed.

MQXR2_PUT_WITH_DEF_ACTION
Put with default action.
MQCXP – ExitResponse2 field

This is set by the receiver’s channel message exit. It indicates that the message is to be put with the MCA’s default action, that is either the MCA’s default user ID, or the context UserIdentifier in the MQMD (message descriptor) of the message.

The value of this constant is zero, which corresponds to the initial value set when the exit is invoked. The constant is provided for documentation purposes.

MQXR2_PUT_WITH_DEF_USERID
Put with default user identifier.

This can only be set by the receiver’s channel message exit. It indicates that the message is to be put with the MCA’s default user identifier.

MQXR2_PUT_WITH_MSG_USERID
Put with message’s user identifier.

This can only be set by the receiver’s channel message exit. It indicates that the message is to be put with the context UserIdentifier in the MQMD (message descriptor) of the message (this may have been modified by the exit).

Only one of MQXR2_PUT_WITH_DEF_ACTION, MQXR2_PUT_WITH_DEF_USERID, and MQXR2_PUT_WITH_MSG_USERID should be set.

MQXR2_USE_AGENT_BUFFER
Use agent buffer.

This indicates that any data to be passed on is in AgentBuffer, not ExitBufferAddr.

The value of this constant is zero, which corresponds to the initial value set when the exit is invoked. The constant is provided for documentation purposes.

MQXR2_USE_EXIT_BUFFER
Use exit buffer.

This indicates that any data to be passed on is in ExitBufferAddr, not AgentBuffer.

Only one of MQXR2_USE_AGENT_BUFFER and MQXR2_USE_EXIT_BUFFER should be set.

MQXR2_DEFAULT_CONTINUATION
Default continuation.

Continuation with the next exit in the chain depends on the response from the last exit invoked:

- If MQXCC_SUPPRESS_FUNCTION or MQXCC_CLOSE_CHANNEL are returned, no further exits in the chain are called.
- Otherwise, the next exit in the chain is invoked.

MQXR2_CONTINUE_CHAIN
Continue with the next exit.

MQXR2_SUPPRESS_CHAIN
Skip remaining exits in chain.

This is an input/output field to the exit.
Feedback (MQLONG)
Feedback code.

This is set to MQFB_NONE on entry to the exit routine.

If a channel message exit sets the ExitResponse field to MQXCC.SuppressFunction, the Feedback field specifies the feedback code that identifies why the message was put on the dead-letter (undelivered-message) queue, and is also used to send an exception report if one has been requested. In this case, if the Feedback field is MQFB_NONE, the following feedback code is used:

MQFB_STOPPED_BY_MSG_EXIT
Message stopped by channel message exit.

The value returned in this field by channel security, send, receive, and message-retry exits is not used by the MCA.

The value returned in this field by auto-definition exits is not used if ExitResponse is MQXCC_OK, but otherwise is used for the AuxErrorDataInt1 parameter in the event message.

This is an input/output field from the exit.

MaxSegmentLength (MQLONG)
Maximum segment length.

This is the maximum length in bytes that can be sent in a single transmission. It is not used by the auto-definition exit. It is of interest to a channel send exit, because this exit must ensure that it does not increase the size of a transmission segment to a value greater than MaxSegmentLength. The length includes the initial 8 bytes that the exit must not change. The value is negotiated between the message channel agents when the channel is initiated. See "Writing and compiling channel-exit programs" on page 585 for more information about segment lengths.

The value in this field is not meaningful if ExitReason is MQXR_INIT.

This is an input field to the exit.

ExitUserArea (MQBYTE16)
Exit user area.

This is a field that is available for the exit to use. (It is not used by the auto-definition exit.) It is initialized to binary zero before the first invocation of the exit (which has an ExitReason set to MQXR_INIT), and thereafter any changes made to this field by the exit are preserved across invocations of the exit.

The following value is defined:

MQXUA_NONE
No user information.

The value is binary zero for the length of the field.

For the C programming language, the constant MQXUA_NONE_ARRAY is also defined; this has the same value as MQXUA_NONE, but is an array of characters instead of a string.
MQCXP – ExitUserArea field

The length of this field is given by MQ_EXIT_USER_AREA_LENGTH. This is an input/output field to the exit.

**ExitData (MQCHAR32)**

Exit data.

This is set on entry to the exit routine to information that the MCA took from the channel definition. If no such information is available, this field is all blanks.

The length of this field is given by MQ_EXIT_DATA_LENGTH.

This is an input field to the exit.

The following fields in this structure are not present if Version is less than MQCXP_VERSION_2.

**MsgRetryCount (MQLONG)**

Number of times the message has been retried.

The first time the exit is invoked for a particular message, this field has the value zero (no retries yet attempted). On each subsequent invocation of the exit for that message, the value is incremented by one by the MCA. On z/OS, the value is always zero.

This is an input field to the exit. The value in this field is not meaningful if ExitReason is MQXR_INIT. The field is not present if Version is less than MQCXP_VERSION_2.

**MsgRetryInterval (MQLONG)**

Minimum interval in milliseconds after which the put operation should be retried.

The first time the exit is invoked for a particular message, this field contains the value of the **MsgRetryInterval** channel attribute. The exit can leave the value unchanged, or modify it to specify a different time interval in milliseconds. If the exit returns MQXCC_OK in ExitResponse, the MCA will wait for at least this time interval before retrying the MQOPEN or MQPUT operation. The time interval specified must be zero or greater.

The second and subsequent times the exit is invoked for that message, this field contains the value returned by the previous invocation of the exit.

If the value returned in the **MsgRetryInterval** field is less than zero or greater than 999 999 999, and ExitResponse is MQXCC_OK, the MCA ignores the **MsgRetryInterval** field in MQCXP and waits instead for the interval specified by the **MsgRetryInterval** channel attribute. On z/OS, the value of this field is always zero.

This is an input/output field to the exit. The value in this field is not meaningful if ExitReason is MQXR_INIT. The field is not present if Version is less than MQCXP_VERSION_2.

**MsgRetryReason (MQLONG)**

Reason code from previous attempt to put the message.

This is the reason code from the previous attempt to put the message; it is one of the MQRC_* values. On z/OS the value of this field is always zero.
This is an input field to the exit. The value in this field is not meaningful if ExitReason is MQXR_INIT. The field is not present if Version is less than MQCXP_VERSION_2.

The following fields in this structure are not present if Version is less than MQCXP_VERSION_3.

**HeaderLength (MQLONG)**
Length of header information.

This field is relevant only for a message exit. The value is the length of the routing header structures at the start of the message data; these are the MQXQH structure, and (for a distribution-list message) the MQDH structure and arrays of MQOR and MQPMR records that follow the MQXQH structure.

The message exit can examine this header information, and modify it if necessary, but the data that the exit returns must still be in the correct format. The exit must not, for example, encrypt or compress the header data at the sending end, even if the message exit at the receiving end makes compensating changes.

If the message exit modifies the header information in such a way as to change its length (for example, by adding another destination to a distribution-list message), it must change the value of HeaderLength correspondingly before returning.

This is an input/output field to the exit. The value in this field is not meaningful if ExitReason is MQXR_INIT. The field is not present if Version is less than MQCXP_VERSION_3.

**PartnerName (MQCHAR48)**
Partner Name.

The name of the partner, as follows:
- For SVRCONN channels, it is the logged-on user ID at the client.
- For all other types of channel, it is the queue-manager name of the partner.

When the exit is initialized this field is blank because the queue manager does not know the name of the partner until after initial negotiation has taken place.

This is an input field to the exit. The field is not present if Version is less than MQCXP_VERSION_3.

**FAPLevel (MQLONG)**
Negotiated Formats and Protocols level.

This is an input field to the exit. The field is not present if Version is less than MQCXP_VERSION_3.

**CapabilityFlags (MQLONG)**
Capability flags.

The following are defined:

**MQCF_NONE**
No flags.

**MQCF_DIST_LISTS**
Distribution lists supported.
MQCXP – CapabilityFlags field

This is an input field to the exit. The field is not present if Version is less than MQCXP_VERSION_3.

ExitNumber (MQLONG)

Exit number.

The ordinal number of the exit, within the type defined in ExitId. For example, if the exit being invoked is the third message exit defined, this field contains the value 3. If the exit type is one for which a list of exits cannot be defined (for example, a security exit), this field has the value 1.

This is an input field to the exit. The field is not present if Version is less than MQCXP_VERSION_3.

The following fields in this structure are not present if Version is less than MQCXP_VERSION_5.

ExitSpace (MQLONG)

Number of bytes in transmission buffer reserved for exit to use.

This field is relevant only for a send exit. It specifies the amount of space in bytes that the MCA will reserve in the transmission buffer for the exit to use. This allows the exit to add to the transmission buffer a small amount of data (typically not exceeding a few hundred bytes) for use by a complementary receive exit at the other end. The data added by the send exit must be removed by the receive exit.

Note: This facility should not be used to send large amounts of data, as this may degrade performance, or even inhibit operation of the channel.

By setting ExitSpace the exit is guaranteed that there will always be at least that number of bytes available in the transmission buffer for the exit to use. However, the exit can use less than the amount reserved, or more than the amount reserved if there is space available in the transmission buffer. The exit space in the buffer is provided following the existing data.

ExitSpace can be set by the exit only when ExitReason has the value MQXR_INIT; in all other cases the value returned by the exit is ignored. On input to the exit, ExitSpace is zero for the MQXR_INIT call, and is the value returned by the MQXR_INIT call in other cases.

If the value returned by the MQXR_INIT call is negative, or there are fewer than 1024 bytes available in the transmission buffer for message data after reserving the requested exit space for all of the send exits in the chain, the MCA outputs an error message and closes the channel. Similarly, if during data transfer the exits in the send exit chain allocate more user space than they reserved such that fewer than 1024 bytes remain in the transmission buffer for message data, the MCA outputs an error message and closes the channel. The limit of 1024 allows the channel’s control and administrative flows to be processed by the chain of send exits, without the need for the flows to be segmented.

This is an input/output field to the exit if ExitReason is MQXR_INIT, and an input field in all other cases. The field is not present if Version is less than MQCXP_VERSION_5.
C declaration

typedef struct tagMQCXP MQCXP;
struct tagMQCXP {
  MQCHAR4 StrucId;   /* Structure identifier */
  MQLONG Version;    /* Structure version number */
  MQLONG ExitId;     /* Type of exit */
  MQLONG ExitReason; /* Reason for invoking exit */
  MQLONG ExitResponse; /* Response from exit */
  MQLONG ExitResponse2; /* Secondary response from exit */
  MQLONG Feedback;  /* Feedback code */
  MQLONG MaxSegmentLength; /* Maximum segment length */
  MQBYTE16 ExitUserArea; /* Exit user area */
  MQCHAR32 ExitData;   /* Exit data */
  MQLONG MsgRetryCount; /* Number of times the message has been retried */
  MQLONG MsgRetryInterval; /* Minimum interval in milliseconds after which the put operation should be retried */
  MQLONG MsgRetryReason; /* Reason code from previous attempt to put the message */
  MQLONG HeaderLength; /* Length of header information */
  MQCHAR48 PartnerName; /* Partner Name */
  MQLONG FAPLevel; /* Negotiated Formats and Protocols level */
  MQLONG CapabilityFlags; /* Capability flags */
  MQLONG ExitNumber; /* Exit number */
  MQLONG ExitSpace;  /* Number of bytes in transmission buffer reserved for exit to use */
};

COBOL declaration

** MQCXP structure
  10 MQCXP.
** Structure identifier
  15 MQCXP-STRUCID PIC X(4).
** Structure version number
  15 MQCXP-VERSION PIC S9(9) BINARY.
** Type of exit
  15 MQCXP-EXITID PIC S9(9) BINARY.
** Reason for invoking exit
  15 MQCXP-EXITREASON PIC S9(9) BINARY.
** Response from exit
  15 MQCXP-EXITRESPONSE PIC S9(9) BINARY.
** Secondary response from exit
  15 MQCXP-EXITRESPONSE2 PIC S9(9) BINARY.
** Feedback code
  15 MQCXP-FEEDBACK PIC S9(9) BINARY.
** Maximum segment length
  15 MQCXP-MAXSEGMENTLENGTH PIC S9(9) BINARY.
** Exit user area
  15 MQCXP-EXITUSERAREA PIC X(16).
** Exit data
  15 MQCXP-EXITDATA PIC X(32).
** Number of times the message has been retried
  15 MQCXP-MSGRETRYCOUNT PIC S9(9) BINARY.
** Minimum interval in milliseconds after which the put operation should be retried
  15 MQCXP-MSGRETRYINTERVAL PIC S9(9) BINARY.
** Reason code from previous attempt to put the message
  15 MQCXP-MSGRETRYREASON PIC S9(9) BINARY.
** Length of header information
  15 MQCXP-HEADERLENGTH PIC S9(9) BINARY.
** Partner Name
  15 MQCXP-PARTNERNAME PIC X(48).
** Negotiated Formats and Protocols level
**MQCXP – Language declarations**

15 MQCXP-FAPLEVEL PIC S9(9) BINARY.
** Capability flags
15 MQCXP-CAPABILITYFLAGS PIC S9(9) BINARY.
** Exit number
15 MQCXP-EXITNUMBER PIC S9(9) BINARY.
** Number of bytes in transmission buffer reserved for exit to use
15 MQCXP-EXITSPACE PIC S9(9) BINARY.

**PL/I declaration**

dcl
1 MQCXP based,
3 StrucId char(4), /* Structure identifier */
3 Version fixed bin(31), /* Structure version number */
3 ExitId fixed bin(31), /* Type of exit */
3 ExitReason fixed bin(31), /* Reason for invoking exit */
3 ExitResponse fixed bin(31), /* Response from exit */
3 ExitResponse2 fixed bin(31), /* Secondary response from exit */
3 Feedback fixed bin(31), /* Feedback code */
3 MaxSegmentLength fixed bin(31), /* Maximum segment length */
3 ExitUserArea char(16), /* Exit user area */
3 ExitData char(32), /* Exit data */
3 MsgRetryCount fixed bin(31), /* Number of times the message has been retried */
3 MsgRetryInterval fixed bin(31), /* Minimum interval in milliseconds after which the put operation should be retried */
3 MsgRetryReason fixed bin(31), /* Reason code from previous attempt to put the message */
3 HeaderLength fixed bin(31), /* Length of header information */
3 PartnerName char(48), /* Partner Name */
3 FAPLevel fixed bin(31), /* Negotiated Formats and Protocols level */
3 CapabilityFlags fixed bin(31), /* Capability flags */
3 ExitNumber fixed bin(31), /* Exit number */
3 ExitSpace fixed bin(31); /* Number of bytes in transmission buffer reserved for exit to use */

**RPG declaration (ILE)**

D*...1....2....3....4....5....6....7...
D* MQCXP Structure
D*
D* Structure identifier
D CXSID 1 4
D* Structure version number
D CXVER 5 8I 0
D* Type of exit
D CXXID 9 12I 0
D* Reason for invoking exit
D CXREA 13 16I 0
D* Response from exit
D CXRES 17 20I 0
D* Secondary response from exit
D CXRE2 21 24I 0
D* Feedback code
D CXFB 25 28I 0
D* Maximum segment length
D CXMSL 29 32I 0
D* Exit user area
D CXUA 33 48
D* Exit data
D CXDAT 49 80
D* Number of times the message has been retried
D CXMRC 81 84I 0
D* Minimum interval in milliseconds after which the put operation
MQCXP – Language declarations

D* should be retried
D CXMRI 85 88I 0
D* Reason code from previous attempt to put the message
D CXMRR 89 92I 0
D* Length of header information
D CXHDL 93 96I 0
D* Partner Name
D CXPNM 97 144
D* Negotiated Formats and Protocols level
D CXFAP 145 148I 0
D* Capability flags
D CXCAP 149 152I 0
D* Exit number
D CXEXN 153 156I 0
D* Number of bytes in transmission buffer reserved for exit to use
D CXHDL 157 160I 0

RPG declaration (OPM)

I*..1.........2........3........4........5........6........7..
I* MQCXP Structure
I*
I* Structure identifier
I 1 4 CXSID
I* Structure version number
I B 5 80CXVER
I* Type of exit
I B 9 120CXXID
I* Reason for invoking exit
I B 13 160CXREA
I* Response from exit
I B 17 200CXRES
I* Secondary response from exit
I B 21 240CXRE2
I* Feedback code
I B 25 280CFB
I* Maximum segment length
I B 29 320CXMSL
I* Exit user area
I 33 48 CXUA
I* Exit data
I 49 80 CXDAT
I* Number of times the message has been retried
I B 81 840CXMRC
I* Minimum interval in milliseconds after which the put operation
I* should be retried
I B 85 880CXMRI
I* Reason code from previous attempt to put the message
I B 89 920CXMRR
I* Length of header information
I B 93 960CXHDL
I* Partner Name
I 97 144 CXPNM
I* Negotiated Formats and Protocols level
I B 145 1480CXFAP
I* Capability flags
I B 149 1520CXCAP
I* Exit number
I B 153 1560CXEXN
I* Number of bytes in transmission buffer reserved for exit to use
I B 157 1600CXHDL

System/390 assembler declaration

<table>
<thead>
<tr>
<th>MQCXP</th>
<th>DSECT</th>
</tr>
</thead>
<tbody>
<tr>
<td>MQCXP_STRUCID</td>
<td>DS CL4</td>
</tr>
<tr>
<td>MQCXP_VERSION</td>
<td>DS F</td>
</tr>
</tbody>
</table>
MQCXP – Language declarations

MQCXP_EXITID   DS   F   Type of exit
MQCXP_EXITREASON   DS   F   Reason for invoking exit
MQCXP_EXITRESPONSE   DS   F   Response from exit
MQCXP_EXITRESPONSE2   DS   F   Secondary response from exit
MQCXP_FEEDBACK   DS   F   Feedback code
MQCXP_MAXSEGMENTLENGTH   DS   F   Maximum segment length
MQCXP_EXITUSERAREA   DS   XL16   Exit user area
MQCXP_EXITDATA   DS   CL32   Exit data
MQCXP_MSGRETRYCOUNT   DS   F   Number of times the message has been retried
MQCXP_MSGRETRYINTERVAL   DS   F   Minimum interval in milliseconds after which the put operation should be retried
MQCXP_MSGRETRYREASON   DS   F   Reason code from previous attempt to put the message
MQCXP_HEADERLENGTH   DS   F   Length of header information
MQCXP_PARTNERNAME   DS   CL48   Partner Name
MQCXP_FAPLEVEL   DS   F   Negotiated Formats and Protocols level
MQCXP_CAPABILITYFLAGS   DS   F   Capability flags
MQCXP_EXITNUMBER   DS   F   Exit number
MQCXP_EXITSPACE   DS   F   Number of bytes in transmission buffer reserved for exit to use
MQCXP_LENGTH   EQU   *-MQCXP

ORG MQCXP
MQCXP_AREA   DS   CL(MQCXP_LENGTH)
MQTXP – Transport exit parameter

The following table summarizes the fields in the structure.

Table 62. Fields in MQTXP

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>StrucId</td>
<td>Structure identifier</td>
<td>681</td>
</tr>
<tr>
<td>Version</td>
<td>Structure version number</td>
<td>681</td>
</tr>
<tr>
<td>ExitReason</td>
<td>Reason for invoking exit</td>
<td>682</td>
</tr>
<tr>
<td>ExitUserArea</td>
<td>Exit user area</td>
<td>682</td>
</tr>
<tr>
<td>TransportType</td>
<td>Transport type</td>
<td>682</td>
</tr>
<tr>
<td>RetryCount</td>
<td>Number of times data has been retried</td>
<td>683</td>
</tr>
<tr>
<td>DataLength</td>
<td>Length of data to be sent</td>
<td>683</td>
</tr>
<tr>
<td>SessionId</td>
<td>Session identifier</td>
<td>683</td>
</tr>
<tr>
<td>GroupId</td>
<td>Group identifier</td>
<td>683</td>
</tr>
<tr>
<td>DataId</td>
<td>Data identifier</td>
<td>683</td>
</tr>
<tr>
<td>ExitResponse</td>
<td>Response from exit</td>
<td>683</td>
</tr>
</tbody>
</table>

The MQTXP structure describes the information that is passed to the transport retry exit.

This structure is supported in the following environments: AIX.

Fields

**StrucId (MQCHAR4)**
Structure identifier.

The value is:

**MQTXP_STRUC_ID**
Identifier for transport retry exit parameter structure.

For the C programming language, the constant **MQTXP_STRUC_ID_ARRAY** is also defined; this has the same value as **MQTXP_STRUC_ID**, but is an array of characters instead of a string.

This is an input field to the exit.

**Version (MQLONG)**
Structure version number.

The value is:

**MQTXP_VERSION_1**
Version-1 transport retry exit parameter structure.

The following constant specifies the version number of the current version:

**MQTXP_CURRENT_VERSION**
Current version of transport retry exit parameter structure.

This is an input field to the exit.
Reserved (MQLONG)
Reserved.

This is a reserved field. The value is zero.

ExitReason (MQLONG)
Reason for invoking exit.

This indicates the reason why the exit is being called. Possible values are:

MQXR_INIT
Exit initialization.
This indicates that the exit is being invoked for the first time. It allows the exit to acquire and initialize any resources that it may need (for example: main storage).

MQXR_TERM
Exit termination.
This indicates that the exit is about to be terminated. The exit should free any resources that it may have acquired since it was initialized (for example: main storage).

MQXR_RETRY
Retry a message.

MQXR_END_BATCH
Called from MCA when batch completed.

MQXR_ACK_RECEIVED
Called from MCA when an acknowledgement has been received.

This is an input field to the exit.

ExitUserArea (MQBYTE16)
Exit user area.

This is a field that is available for the exit to use. It is initialized to MQXUA_NONE (binary zero) before the first invocation of the exit, and thereafter any changes made to this field by the exit are preserved across invocations of the exit. The first invocation of the exit has ExitReason set to MQXR_INIT.

The following value is defined:

MQXUA_NONE
No user information.
The value is binary zero for the length of the field.
For the C programming language, the constant MQXUA_NONE_ARRAY is also defined; this has the same value as MQXUA_NONE, but is an array of characters instead of a string.

The length of this field is given by MQ_EXIT_USER_AREA_LENGTH. This is an input/output field to the exit.

TransportType (MQLONG)
Transport type.

This is the type of transport being used. The value is:
MQXPT_UDP
    UDP transport protocol.

This is an input field to the exit.

**RetryCount (MQLONG)**
Number of times data has been retried.

This is the number of previous attempts that have been made to send the current data. It is zero on first invocation of the exit for the current data.

This is an input field to the exit.

**DataLength (MQLONG)**
Length of data to be sent.

This is always greater than zero. For MQXPT_UDP, it is one complete encoded datagram.

This is an input field to the exit.

**SessionId (MQLONG)**
Session identifier.

This is the identifier of the session of channel. For MQXPT_UDP, it is the UdpHandle.

This is an input field to the exit.

**GroupId (MQLONG)**
Group identifier.

This is the identifier of the group, bunch, or message to which the data belongs. For MQXPT_UDP, it identifies the bunch.

This is an input field to the exit.

**DataId (MQLONG)**
Data identifier.

For MQXPT_UDP, this is the datagram identifier.

This is an input field to the exit.

**ExitResponse (MQLONG)**
Response from exit.

This is set by the exit to indicate how processing should continue. It must be one of the following:

**MQXCC_OK**
Exit completed successfully.

This indicates that processing should continue normally.

**MQXCC_REQUEST_ACK**
Request acknowledgement.
MQTXP – ExitResponse field

This indicates that processing should continue normally, but that the datagram about to be sent should request that an acknowledgement be returned by the receiver of the datagram.

MQXCC_CLOSE_CHANNEL
Close channel.

This indicates that processing should be discontinued and the channel closed.

If any other value is returned by the exit, processing continues as if MQXCC_CLOSE_CHANNEL had been specified.

This is an output field from the exit.

Feedback (MQLONG)
Reserved.

This is a reserved field. The value is zero.

C declaration

typedef struct tagMQTXP MQTXP;
struct tagMQTXP {
MQCHAR4 StrucId; /* Structure identifier */
MQLONG Version; /* Structure version number */
MQLONG Reserved; /* Reserved */
MQLONG ExitReason; /* Reason for invoking exit */
MQLONG ExitUserArea; /* Exit user area */
MQLONG TransportType; /* Transport type */
MQLONG RetryCount; /* Number of times data has been retried */
MQLONG DataLength; /* Length of data to be sent */
MQLONG SessionId; /* Session identifier */
MQLONG GroupId; /* Group identifier */
MQLONG DataId; /* Data identifier */
MQLONG ExitResponse; /* Response from exit */
MQLONG Feedback; /* Reserved */
};
MQXWD – Exit wait descriptor

The following table summarizes the fields in the structure.

Table 63. Fields in MQXWD

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>StrucId</td>
<td>Structure identifier</td>
<td>685</td>
</tr>
<tr>
<td>Version</td>
<td>Structure version number</td>
<td>685</td>
</tr>
<tr>
<td>ECB</td>
<td>Event control block to wait on</td>
<td>686</td>
</tr>
</tbody>
</table>

The MQXWD structure is an input/output parameter on the MQXWAIT call.

This structure is supported only on z/OS.

Fields

**StrucId (MQCHAR4)**
Structure identifier.

The value must be:

MQXWD_STRUC_ID
Identifier for exit wait descriptor structure.

For the C programming language, the constant MQXWD_STRUC_ID_ARRAY is also defined; this has the same value as MQXWD_STRUC_ID, but is an array of characters instead of a string.

The initial value of this field is MQXWD_STRUC_ID.

**Version (MQLONG)**
Structure version number.

The value must be:

MQXWD_VERSION_1
Version number for exit wait descriptor structure.

The initial value of this field is MQXWD_VERSION_1.

**Reserved1 (MQLONG)**
Reserved.

This is a reserved field; its value must be zero.

This is an input field.

**Reserved2 (MQLONG)**
Reserved.

This is a reserved field; its value must be zero.

This is an input field.
MQXWD – Reserved3 field

**Reserved3 (MQLONG)**

Reserved.

This is a reserved field; its value must be zero.

This is an input field.

**ECB (MQLONG)**

Event control block to wait on.

This is the event control block (ECB) to wait on. It should be set to zero before the MQXWAIT call is issued; on successful completion it will contain the post code.

This is an input/output field.

**C declaration**

```c
typedef struct tagMQXWD MQXWD;
struct tagMQXWD {
    MQCHAR4 StrucId; /* Structure identifier */
    MQLONG Version; /* Structure version number */
    MQLONG Reserved1; /* Reserved */
    MQLONG Reserved2; /* Reserved */
    MQLONG Reserved3; /* Reserved */
    MQLONG ECB; /* Event control block to wait on */
};
```

**System/390 assembler declaration**

```assembly
MQXWD DSECT
MQXWD_STRUCID DS CL4 Structure identifier
MQXWD_VERSION DS F Structure version number
MQXWD_RESERVED1 DS F Reserved
MQXWD_RESERVED2 DS F Reserved
MQXWD_RESERVED3 DS F Reserved
MQXWD_ECB DS F Event control block to wait on
*
MQXWD_LENGTH EQU *-MQXWD
ORG MQXWD
MQXWD_AREA DS CL(MQXWD_LENGTH)
```
Chapter 47. Problem determination in DQM

This chapter explains the various aspects of problem determination and suggests methods of resolving problems. Some of the problems mentioned in this chapter are platform and installation specific. Where this is the case, it is made clear in the text.

Problem determination for the following scenarios is discussed:
- “Error message from channel control”
- “Ping”
- “Dead-letter queue considerations” on page 688
- “Validation checks” on page 688
- “In-doubt relationship” on page 689
- “Channel startup negotiation errors” on page 689
- “When a channel refuses to run” on page 689
- “Retriving the link” on page 692
- “Data structures” on page 692
- “User exit problems” on page 692
- “Disaster recovery” on page 693
- “Channel switching” on page 693
- “Connection switching” on page 694
- “Client problems” on page 694
- “Error logs” on page 694

Error message from channel control

Problems found during normal operation of the channels are reported to the system console and to the system log. In WebSphere MQ for z/OS using CICS, they are reported to the CICS Transient Data Queue CKMQ, if that is defined and available. In WebSphere MQ for Windows they are reported to the channel log. Problem diagnosis starts with the collection of all relevant information from the log, and analysis of this information to identify the problem.

However, this could be difficult in a network where the problem may arise at an intermediate system that is staging some of your messages. An error situation, such as transmission queue full, followed by the dead-letter queue filling up, would result in your channel to that site closing down.

In this example, the error message you receive in your error log will indicate a problem originating from the remote site, but may not be able to tell you any details about the error at that site.

You need to contact your counterpart at the remote site to obtain details of the problem, and to receive notification of that channel becoming available again.

Ping

Ping is useful in determining whether the communication link and the two message channel agents that make up a message channel are functioning across all interfaces.

Ping makes no use of transmission queues, but it does invoke some user exit programs. If any error conditions are encountered, error messages are issued.
Ping

To use ping, you can issue the MQSC command PING CHANNEL (you cannot do this if you are using CICS for distributed queuing on z/OS). On z/OS and iSeries, you can also use the panel interface to select this option.

On UNIX platforms, OS/2, Windows, and iSeries, you can also use the MQSC command PING QMGR to test whether the queue manager is responsive to commands. See the WebSphere MQ Script (MQSC) Command Reference for more information about this.

Dead-letter queue considerations

In some WebSphere MQ products the dead-letter queue is referred to as an undelivered-message queue.

If a channel ceases to run for any reason, applications will probably continue to place messages on transmission queues, creating a potential overflow situation. Applications can monitor transmission queues to find the number of messages waiting to be sent, but this would not be a normal function for them to carry out.

When this occurs in a message-originating node, and the local transmission queue is full, the application’s PUT fails.

When this occurs in a staging or destination node, there are three ways that the MCA copes with the situation:

1. By calling the message-retry exit, if one is defined.
2. By directing all overflow messages to a dead-letter queue (DLQ), returning an exception report to applications that requested these reports.

   Note: In distributed-queuing management, if the message is too big for the DLQ, the DLQ is full, or the DLQ is not available, the channel stops and the message remains on the transmission queue. Ensure your DLQ is defined, available, and sized for the largest messages you handle.

3. By closing down the channel, if neither of the previous options succeeded.
4. By returning the undelivered messages back to the sending end and returning a full report to the reply-to queue (MQRC_EXCEPTION_WITH_FULL_DATA and MQRO_DISCARD_MSG).

If an MCA is unable to put a message on the DLQ:

• The channel stops
• Appropriate error messages are issued at the system consoles at both ends of the message channel
• The unit of work is backed out, and the messages reappear on the transmission queue at the sending channel end of the channel
• Triggering is disabled for the transmission queue

Validation checks

A number of validation checks are made when creating, altering, and deleting channels, and where appropriate, an error message returned.

Errors may occur when:

• A duplicate channel name is chosen when creating a channel
• Unacceptable data is entered in the channel parameter fields
• The channel to be altered is in doubt, or does not exist
In-doubt relationship

If a channel is in doubt, it is usually resolved automatically on restart, so the system operator does not need to resolve a channel manually in normal circumstances. See "In-doubt channels" on page 70 for information about this.

Channel startup negotiation errors

During channel startup, the starting end has to state its position and agree channel running parameters with the corresponding channel. It may happen that the two ends cannot agree on the parameters, in which case the channel closes down with error messages being issued to the appropriate error logs.

When a channel refuses to run

If a channel refuses to run:

- Check that DQM and the channels have been set up correctly. This is a likely problem source if the channel has never run. Reasons could be:
  - A mismatch of names between sending and receiving channels (remember that uppercase and lowercase letters are significant)
  - Incorrect channel types specified
  - The sequence number queue (if applicable) is not available, or is damaged
  - The dead-letter queue is not available
  - The sequence number wrap value is different on the two channel definitions
  - A queue manager, CICS system, or communication link is not available
  - Following a restart, the wrong queue manager may have been attached to CICS
  - A receiver channel might be in STOPPED state
  - The connection might not be defined correctly
  - There might be a problem with the communications software (for example, is TCP running?)
  - In z/OS using CICS, check that the DFHSIT SYSSIDNT name of the target CICS system matches the connection name that you have specified for that system
- It is possible that an in-doubt situation exists, if the automatic synchronization on startup has failed for some reason. This is indicated by messages on the system console, and the status panel may be used to show channels that are in doubt.

The possible responses to this situation are:

- Issue a Resolve channel request with Backout or Commit.
  You need to check with your remote link supervisor to establish the number of the last message or unit of work committed. Check this against the last number at your end of the link. If the remote end has committed a number, and that number is not yet committed at your end of the link, then issue a RESOLVE COMMIT command.

  In all other cases, issue a RESOLVE BACKOUT command.

  The effect of these commands is that backed out messages reappear on the transmission queue and are sent again, while committed messages are discarded.

  If in doubt yourself, perhaps backing out with the probability of duplicating a sent message would be the safer decision.
Channel refuses to run

- Issue a RESET command.

This command is for use when sequential numbering is in effect, and should be used with care. Its purpose is to reset the sequence number of messages and you should use it only after using the RESOLVE command to resolve any in-doubt situations.

- On WebSphere MQ for iSeries, Windows, UNIX systems, and z/OS without CICS, and MQSeries for OS/2 Warp, there is no need for the administrator to choose a particular sequence number to ensure that the sequence numbers are put back in step. When a sender channel starts up after being reset, it informs the receiver that it has been reset and supplies the new sequence number that is to be used by both the sender and receiver.

**Note:** If the sender is WebSphere MQ for z/OS using CICS, the sequence number should be reset to the same number as any receiving queue managers.

- If the status of a receiver end of the channel is STOPPED, it can be reset by starting the receiver end.

**Note:** This does not start the channel, it merely resets the status. The channel must still be started from the sender end.

### Triggered channels

If a triggered channel refuses to run, the possibility of in-doubt messages should be investigated as described above.

Another possibility is that the trigger control parameter on the transmission queue has been set to NOTRIGGER by the channel. This happens when:

- There is a channel error
- The channel was stopped because of a request from the receiver
- The channel was stopped because of a problem on the sender that requires manual intervention

After diagnosing and fixing the problem, you must start the channel manually.

An example of a situation where a triggered channel fails to start is as follows:

1. A transmission queue is defined with a trigger type of FIRST.
2. A message arrives on the transmission queue, and a trigger message is produced.
3. The channel is started, but stops immediately because the communications to the remote system are not available.
4. The remote system is made available.
5. Another message arrives on the transmission queue.
6. The second message does not increase the queue depth from zero to one, so no trigger message is produced (unless the channel is in RETRY state). If this happens, the channel must be started manually.

On WebSphere MQ for z/OS, if the queue manager is stopped using MODE(FORCE) during channel initiator shutdown, it may be necessary to manually restart some channels after channel initiator restart.
Conversion failure

Another reason for the channel refusing to run could be that neither end is able to carry out necessary conversion of message descriptor data between ASCII and EBCDIC, and integer formats. In this instance, communication is not possible.

Network problems

When using LU 6.2, make sure that your definitions are consistent throughout the network. For example, if you have increased the RU sizes in your CICS Transaction Server for z/OS or Communications Manager definitions, but you have a controller with a small MAXDATA value in its definition, the session may fail if you attempt to send large messages across the network. A symptom of this may be that channel negotiation takes place successfully, but the link fails when message transfer occurs.

When using TCP, if your channels are unreliable and your connections break, set a KEEPALIVE value for your system or channels. You can use the SO_KEEPALIVE option to set a system-wide value, and on WebSphere MQ for z/OS, you can also use the KeepAlive Interval channel attribute (KAINT) to set channel-specific keepalive values. These options are discussed in “Checking that the other end of the channel is still available” on page 66 and “KeepAlive Interval (KAINT)” on page 87.

Adopting an MCA

The Adopt MCA function enables WebSphere MQ to cancel a receiver channel and to start a new one in its place.

For more information about this function, see “Adopting an MCA” on page 67. For details of its parameters, see WebSphere MQ for z/OS System Setup Guide.

Registration time for DDNS

When a group TCP/IP listener is started, it registers with DDNS. But there may be a delay until the address is available to the network. A channel that is started in this period, and which targets the newly registered generic name, fails with an ‘error in communications configuration’ message. The channel then goes into retry until the name becomes available to the network. The length of the delay will be dependent on the name server configuration used.

Dial-up problems

WebSphere MQ supports connection over dial-up lines but you should be aware that with TCP, some protocol providers assign a new IP address each time you dial in. This can cause channel synchronization problems because the channel cannot recognize the new IP addresses and so cannot ensure the authenticity of the partner. If you encounter this problem, you need to use a security exit program to override the connection name for the session.

This problem does not occur when a WebSphere MQ for AIX, iSeries, HP-UX, Linux, Solaris, and Windows, or MQSeries V5.1 for Compaq Tru64 UNIX, and OS/2 Warp product is communicating with another product at the same level, because the queue manager name is used for synchronization instead of the IP address.
Retrying the link

An error scenario may occur that is difficult to recognize. For example, the link and channel may be functioning perfectly, but some occurrence at the receiving end causes the receiver to stop. Another unforeseen situation could be that the receiver system has run out of storage and is unable to complete a transaction.

You need to be aware that such situations can arise, often characterized by a system that appears to be busy but is not actually moving messages. You need to work with your counterpart at the far end of the link to help detect the problem and correct it.

Retry considerations

If a link failure occurs during normal operation, a sender or server channel program will itself start another instance, provided that:
1. Initial data negotiation and security exchanges are complete
2. The retry count in the channel definition is greater than zero

Note: For OS/2, iSeries, UNIX systems, and Windows, to attempt a retry a channel initiator must be running. In platforms other than WebSphere MQ for AIX, iSeries, HP-UX, Linux, Solaris, and Windows, and MQSeries V5.1 for Compaq Tru64 UNIX, and OS/2 Warp, this channel initiator must be monitoring the initiation queue specified in the transmission queue that the channel is using.

Shared channel recovery on z/OS

See “Shared channel recovery” on page 442, which includes a table that shows the types of shared-channel failure and how each type is handled.

Data structures

Data structures are needed for reference when checking logs and trace entries during problem diagnosis. Details can be found in Chapter 46, “Channel-exit calls and data structures” on page 611 and in the WebSphere MQ Application Programming Reference book.

User exit problems

The interaction between the channel programs and the user-exit programs has some error-checking routines, but this facility can only work successfully when the user exits obey the rules described in Part 7, “Further intercommunication considerations” on page 567. When errors occur, the most likely outcome will be that the channel stops and the channel program issues an error message, together with any return codes from the user exit. Any errors detected on the user exit side of the interface can be determined by scanning the messages created by the user exit itself.

You might need to use a trace facility of your host system to identify the problem.
Disaster recovery planning is the responsibility of individual installations, and the functions performed may include the provision of regular system ‘snapshot’ dumps that are stored safely off-site. These dumps would be available for regenerating the system, should some disaster overtake it. If this occurs, you need to know what to expect of the messages, and the following description is intended to start you thinking about it.

First a recap on system restart. If a system fails for any reason, it may have a system log that allows the applications running at the time of failure to be regenerated by replaying the system software from a syncpoint forward to the instant of failure. If this occurs without error, the worst that can happen is that message channel syncpoints to the adjacent system may fail on startup, and that the last batches of messages for the various channels will be sent again. Persistent messages will be recovered and sent again, nonpersistent messages may be lost.

If the system has no system log for recovery, or if the system recovery fails, or where the disaster recovery procedure is invoked, the channels and transmission queues may be recovered to an earlier state, and the messages held on local queues at the sending and receiving end of channels may be inconsistent.

Messages may have been lost that were put on local queues. The consequence of this happening depends on the particular WebSphere MQ implementation, and the channel attributes. For example, where strict message sequencing is in force, the receiving channel detects a sequence number gap, and the channel closes down for manual intervention. Recovery then depends upon application design, as in the worst case the sending application may need to restart from an earlier message sequence number.

Channel switching

A possible solution to the problem of a channel ceasing to run would be to have two message channels defined for the same transmission queue, but with different communication links. One message channel would be preferred, the other would be a replacement for use when the preferred channel is unavailable.

If triggering is required for these message channels, the associated process definitions must exist for each sender channel end.

To switch message channels:

• If the channel is triggered, set the transmission queue attribute NOTRIGGER.
• Ensure the current channel is inactive.
• Resolve any in-doubt messages on the current channel.
• If the channel is triggered, change the process attribute in the transmission queue to name the process associated with the replacement channel.
  In this context, some implementations allow a channel to have a blank process object definition, in which case you may omit this step as the queue manager will find and start the appropriate process object.
• Restart the channel, or if the channel was triggered, set the transmission queue attribute TRIGGER.
Connection switching

Another solution would be to switch communication connections from the transmission queues.

To do this:
- If the sender channel is triggered, set the transmission queue attribute NOTRIGGER.
- Ensure the channel is inactive.
- Resolve any in-doubt messages on the channel.
- Change the connection and profile fields to connect to the replacement communication link.
- Ensure that the corresponding channel at the remote end has been defined.
- Restart the channel, or if the sender channel was triggered, set the transmission queue attribute TRIGGER.

Client problems

A client application may receive an unexpected error return code, for example:
- Queue manager not available
- Queue manager name error
- Connection broken

Look in the client error log for a message explaining the cause of the failure. There may also be errors logged at the server, depending on the nature of the failure.

Terminating clients

Even though a client has terminated, it is still possible for its surrogate process to be holding its queues open. Normally this will only be for a short time until the communications layer notifies that the partner has gone.

Error logs

WebSphere MQ error messages are placed in different error logs depending on the platform. There are error logs for:
- OS/2 and Windows
- UNIX systems
- VSE/ESA
- DOS, Windows 3.1, Windows 95, and Windows 98 clients
- z/OS
- MQSeries for Compaq NonStop Kernel

Error logs for OS/2, Windows and Windows 98 client

MQSeries for OS/2 Warp and WebSphere MQ for Windows use a number of error logs to capture messages concerning the operation of WebSphere MQ itself, any queue managers that you start, and error data coming from the channels that are in use.

The location the error logs are stored in depends on whether the queue manager name is known and whether the error is associated with a client.
- If the queue manager name is known and the queue manager is available:
  C:\MQM\QMGRS\QMgrName\ERRORS\AMQERR01.LOG
- If the queue manager is not available:
Error logs

C:\MQM\QMGRS\@SYSTEM\ERRORS\AMQERR01.LOG

- If an error has occurred with a client application:
  C:\MQM\ERRORS\AMQERR01.LOG

Note: The above examples assume that you have installed WebSphere MQ on the C: drive and in the MQM directory. On Windows, the default data path is C:\WINNT\Profiles\All Users\Application Data\MQSeries\.

On Windows, you should also examine the Windows application event log for relevant messages.

Error logs on UNIX systems

WebSphere MQ on UNIX systems uses a number of error logs to capture messages concerning the operation of WebSphere MQ itself, any queue managers that you start, and error data coming from the channels that are in use. The location the error logs are stored in depends on whether the queue manager name is known and whether the error is associated with a client.

- If the queue manager name is known and the queue manager is available:
  /var/mqm/qmgrs/QMgrName/errors/AMQERR01.LOG

- If the queue manager is not available:
  /var/mqm/qmgrs/@SYSTEM/errors/AMQERR01.LOG

- If an error has occurred with a client application:
  /var/mqm/errors/AMQERR01.LOG

Error logs on DOS, Windows 3.1, and Windows 95 clients

WebSphere MQ clients use two error logs, stored in a location set by the environment variable MQDATA (the default is the root drive of the client).

- Error messages:
  AMQERR01.LOG

- FFDC messages:
  AMQERR01.FDC

These files are not readable. See the WebSphere MQ Clients book for information about formatting the information.

Error logs on z/OS

If you are not using CICS, error messages are written to:

- The z/OS system console
- The channel-initiator job log

If you are using the z/OS message processing facility to suppress messages, the console messages may be suppressed. See the WebSphere MQ for z/OS Concepts and Planning Guide for more information.

If you are using CICS, error messages are written to the z/OS system console or the CKMQ extrapartition transient data queue. See the WebSphere MQ for z/OS Concepts and Planning Guide for more information.

Error logs on MQSeries for VSE/ESA

All WebSphere MQ-generated error messages are written to SYSTEM.LOG.
Error logs

Error logs on MQSeries for Compaq NonStop Kernel

For information about this, see “Queue manager configuration file” on page 74.
Part 8. Appendixes
Appendix A. Channel planning form

The form shown in Table 64 on page 701 is supplied for you to create and maintain a list of all message channels for each queue manager in your system. Do not fill in the form in this book. Instead, photocopy it as many times as required to hold the definitions of all the channels in your system. The filled-in form, see Table 65 on page 702, is included to illustrate how the two examples in Chapter 32, “Message channel planning example for z/OS using CICS” on page 431 and Chapter 43, “Message channel planning example for iSeries” on page 541, could be shown.

How to use the form

The channel planning form allows you to keep an overview of the channels and associated objects in your system. It will help to prevent you from making errors when changing your channel configuration.

One of the more obvious errors is to allocate items more than once:

Communications connections identifiers
Allocate only once. It may be possible to share connections between channels when using LU 6.2.

Channel names
Allocate only once.

Transmission queues
Allocate to only one channel. It is possible to allocate to more than one channel for standby purposes, but ensure that only one is active, unless the host environment is WebSphere MQ for z/OS, and there is no sequential delivery of messages selected.

Remote queue definition
The name must be unique.

Queue manager alias name
The name must be unique.

Reply-to queue name
The name must be unique.

Reply-to queue alias name
The name must be unique.

Adjacent channel system name
The name must be unique.

One method of completing the form would be to allocate, systematically, in this order:
• Channels to adjacent systems
• Transmission queues to channels
• Remote queue definitions to queue names and queue manager names, and to transmission queues
• Reply-to queue aliases to reply-to queue names and route names
• Queue manager aliases to remote queue managers and transmission queues
Channel planning form

Proceed as follows:

1. Start with one adjacent system, define the first outward channel to that system, and give it a name.

2. Fill in the channel name on the form with the channel type, transmission queue name, adjacent system name, and remote queue manager name.

3. For each class-of-service, logically-named connection, fill in the logical queue manager name to list the queue manager name resolutions using this channel.

4. Allocate a communication connection and fill in the name and profile, where applicable.

5. Record the names of all the queues that your applications are going to use on this channel, using the columns provided on the form. This is necessary where remote queue definitions are used, so that the name resolutions are listed.

6. Do not forget to include the reply-to alias queue names in this list.

7. Move to the next channel and continue until all outward channels have been completed for this adjacent system.

8. When this has been completed, repeat from the beginning for incoming channels from this adjacent system.

9. Move on to the next adjacent system, and repeat.

10. Check the complete list for unwanted multiple assignments of names, objects and connections.

When the list is complete and checked out, use it as an aid in creating the objects, and defining the channels listed.
<table>
<thead>
<tr>
<th>Channel name</th>
<th>Channel type</th>
<th>CICS system ID (where needed)</th>
<th>Transmission queue name</th>
<th>Connection name</th>
<th>Profile, or mode name</th>
<th>Adjacent system name</th>
<th>Logical queue manager name</th>
<th>Logical queue name</th>
<th>Physical queue manager name</th>
<th>Physical queue name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Channel name</td>
<td>Channel type</td>
<td>CICS system ID (where needed)</td>
<td>Transmission queue name</td>
<td>Connection name</td>
<td>Profile, or mode, name</td>
<td>Adjacent system name</td>
<td>Logical queue manager name</td>
<td>Logical queue name</td>
<td>Physical queue manager name</td>
<td>Physical queue name</td>
</tr>
<tr>
<td>-----------------</td>
<td>--------------</td>
<td>-------------------------------</td>
<td>-------------------------</td>
<td>-----------------</td>
<td>------------------------</td>
<td>----------------------</td>
<td>---------------------------</td>
<td>------------------</td>
<td>-----------------------------</td>
<td>------------------</td>
</tr>
<tr>
<td>QM1.to.QM2</td>
<td>SENDER</td>
<td>(default)</td>
<td>QM2</td>
<td>QM2C</td>
<td>(none)</td>
<td>QM2</td>
<td>QM2</td>
<td>Payrollr</td>
<td>QM2</td>
<td>Payroll</td>
</tr>
<tr>
<td>QM2.to.QM1</td>
<td>RECEIVER</td>
<td>(none)</td>
<td>(none)</td>
<td>(none)</td>
<td>(none)</td>
<td>(none)</td>
<td>(none)</td>
<td>(none)</td>
<td>(none)</td>
<td>(none)</td>
</tr>
</tbody>
</table>

Table 65. Channel planning form. System name: QM2 Queue manager name: QM2 Page no: 1
Appendix B. Constants for channels and exits

This appendix specifies the values of the named constants that apply to channels and exits.

The constants are grouped according to the parameter or field to which they relate. All of the names of the constants in a group begin with a common prefix of the form “MQxxxxx_”, where xxxx represents a string of 0 through 5 characters that indicates the parameter or field to which the values relate. The constants are ordered alphabetically by this prefix.

Notes:
1. For constants with numeric values, the values are shown in both decimal and hexadecimal forms.
2. Hexadecimal values are represented using the notation X'hhhh', where each “h” denotes a single hexadecimal digit.
3. Character values are shown delimited by single quotation marks; the quotation marks are not part of the value.
4. Blanks in character values are represented by one or more occurrences of the symbol “b”.
5. If the value is shown as “(variable)”, it indicates that the value of the constant depends on the environment in which the application is running.

List of constants

The following sections list all of the named constants that are mentioned in this book, and show their values.

**MQ_* (Lengths of character string and byte fields)**

<table>
<thead>
<tr>
<th>Constant</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>MQ_CHANNEL_DESC_LENGTH</td>
<td>64</td>
</tr>
<tr>
<td>MQ_CHANNEL_NAME_LENGTH</td>
<td>20</td>
</tr>
<tr>
<td>MQ_CONN_NAME_LENGTH</td>
<td>264</td>
</tr>
<tr>
<td>MQ_EXIT_DATA_LENGTH</td>
<td>32</td>
</tr>
<tr>
<td>MQ_EXIT_NAME_LENGTH</td>
<td>(variable)</td>
</tr>
<tr>
<td>MQ_LOCAL_ADDRESS_LENGTH</td>
<td>48</td>
</tr>
<tr>
<td>MQ_MAX_EXIT_NAME_LENGTH</td>
<td>128</td>
</tr>
<tr>
<td>MQ_MAX_MCA_USER_ID_LENGTH</td>
<td>64</td>
</tr>
<tr>
<td>MQ_MCA_NAME_LENGTH</td>
<td>20</td>
</tr>
<tr>
<td>MQ_MCA_USER_ID_LENGTH</td>
<td>(variable)</td>
</tr>
<tr>
<td>MQ_MODE_NAME_LENGTH</td>
<td>8</td>
</tr>
<tr>
<td>MQ_PASSWORD_LENGTH</td>
<td>12</td>
</tr>
<tr>
<td>MQ_Q_MGR_NAME_LENGTH</td>
<td>48</td>
</tr>
<tr>
<td>MQ_Q_NAME_LENGTH</td>
<td>48</td>
</tr>
<tr>
<td>MQ_SECURITY_ID_LENGTH</td>
<td>40</td>
</tr>
<tr>
<td>MQ_SHORT_CONN_NAME_LENGTH</td>
<td>20</td>
</tr>
<tr>
<td>MQ_SSL_CIPHER_SPEC_LENGTH</td>
<td>32</td>
</tr>
<tr>
<td>MQ_TOTAL_EXIT_DATA_LENGTH</td>
<td>999</td>
</tr>
<tr>
<td>MQ_TOTAL_EXIT_NAME_LENGTH</td>
<td>999</td>
</tr>
<tr>
<td>MQ_TP_NAME_LENGTH</td>
<td>64</td>
</tr>
<tr>
<td>MQ_USER_ID_LENGTH</td>
<td>12</td>
</tr>
</tbody>
</table>

© Copyright IBM Corp. 1993, 2002
MQCC_* (Completion code)

MQCC_OK 0 X'00000000'
MQCC_FAILED 2 X'00000002'

MQCD_* (Channel definition structure length)

See the StrucLength field described in “MQCD – Channel definition” on page 626

MQCD_LENGTH_4 (variable)
MQCD_LENGTH_5 (variable)
MQCD_LENGTH_6 (variable)
MQCD_LENGTH_7 (variable)
MQCD_CURRENT_LENGTH (variable)

MQCD_* (Channel definition structure version)

See the Version field described in “MQCD – Channel definition” on page 626

MQCD_VERSION_1 1 X'00000001'
MQCD_VERSION_2 2 X'00000002'
MQCD_VERSION_3 3 X'00000003'
MQCD_VERSION_4 4 X'00000004'
MQCD_VERSION_5 5 X'00000005'
MQCD_VERSION_6 6 X'00000006'
MQCD_VERSION_7 7 X'00000007'
MQCD_CURRENT_VERSION (variable)

MQCDC_* (Channel data conversion)

See the DataConversion field described in “MQCD – Channel definition” on page 626

MQCDC_NO_SENDER_CONVERSION 0 X'00000000'
MQCDC_SENDER_CONVERSION 1 X'00000001'

MQCF_* (Channel capability flags)

See the CapabilityFlags field described in “MQCX – Channel exit parameter” on page 666

MQCF_NONE 0 X'00000000'
MQCF_DIST_LISTS 1 X'00000001'

MQCHT_* (Channel type)

See the ChannelType field described in “MQCD – Channel definition” on page 626
MQCXP_* (Channel-exit parameter structure identifier)

See the StrucId field described in “MQCXP – Channel exit parameter” on page 666.

MQCXP_STRUC_ID 'CXPb'

For the C programming language, the following array version is also defined:

MQCXP_STRUC_ID_ARRAY 'C','X','P','b'

MQCXP_* (Channel-exit parameter structure version)

See the Version field described in “MQCXP – Channel exit parameter” on page 666.

MQCXP_VERSION_1 1 X'00000001'
MQCXP_VERSION_2 2 X'00000002'
MQCXP_VERSION_3 3 X'00000003'
MQCXP_VERSION_4 4 X'00000004'
MQCXP_VERSION_5 5 X'00000005'
MQCXP_CURRENT_VERSION (variable)

MQFB_* (Feedback)

See the Feedback field described in “MQCXP – Channel exit parameter” on page 666.

MQFB_NONE 0 X'00000000'
MQFB_STOPPED_BY_MSG_EXIT 268 X'0000010C'

MQKAI_* (Keepalive interval)

MQKAI_AUTO -1 X'FFFFFFFF'

MQMCAT_* (MCA type)

See the MCAType field described in “MQCD – Channel definition” on page 626.

MQMCAT_PROCESS 1 X'00000001'
MQMCAT_THREAD 2 X'00000002'
Constants

MQNPMS_* (Nonpersistent message speed)

See the NonPersistentMsgSpeed field described in “MQCD – Channel definition” on page 626.

MQNPMS_NORMAL 1 X'00000001'
MQNPMS_FAST 2 X'00000002'

MQPA_* (Put authority)

See the PutAuthority field described in “MQCD – Channel definition” on page 626.

MQPA_DEFAULT 1 X'00000001'
MQPA_CONTEXT 2 X'00000002'

MQRC_* (Reason code)

See the Reason parameter described in “MQXWAIT – Wait in exit” on page 624.

MQRC_NONE 0 X'00000000'
MQRC_OPTIONS_ERROR 2046 X'000007FE'
MQRC_PUT_INHIBITED 2051 X'00000803'
MQRC_Q_FULL 2053 X'00000805'
MQRC_XWAIT_CANCELED 2107 X'0000083B'
MQRC_XWAIT_ERROR 2108 X'0000083C'
MQRC_PAGESET_FULL 2192 X'00000890'
MQRC_ADAPTER_NOT_AVAILABLE 2204 X'0000089C'

MQSCA_* (SSL client authentication)

See the SSLClientAuth field described in “MQCD – Channel definition” on page 626.

MQSCA_REQUIRED 0 X'00000000'
MQSCA_OPTIONAL 1 X'00000001'

MQSID_* (Security identifier)

See the MCASecurityId and RemoteSecurityId fields described in “MQCD – Channel definition” on page 626.

MQSID_NONE X'00...00' (40 nulls)

For the C programming language, the following array version is also defined:

MQSID_NONE_ARRAY '\0','\0','...','\0','\0'
MQSIDT_* (Security identifier type)

See the MCASecurityId and RemoteSecurityId fields described in “MQCD – Channel definition” on page 626.

MQSIDT_NONE
X'00'
MQSIDT_NT_SECURITY_ID
X'01'

MQTXP_* (Transport retry exit structure identifier)

See the StrucId field described in “MQTXP – Transport exit parameter” on page 681.

MQTXP_STRUC_ID 'TXPb'

For the C programming language, the following array version is also defined:

MQTXP_STRUC_ID_ARRAY 'T', 'X', 'P', 'b'

MQTXP_* (Transport retry exit structure version)

See the Version field described in “MQTXP – Transport exit parameter” on page 681.

MQTXP_VERSION_1 1 X'00000001'
MQTXP_CURRENT_VERSION 1 X'00000001'

MQXCC_* (Exit response)

See the ExitResponse field described in “MQCXP – Channel exit parameter” on page 666.

MQXCC_REQUEST_ACK -7 X'FFFFFFF9'
MQXCC_CLOSE_CHANNEL -6 X'FFFFFFFA'
MQXCC_SUPPRESS_EXIT -5 X'FFFFFFFB'
MQXCC_SEND_SEC_MSG -4 X'FFFFFFFC'
MQXCC_SEND_AND_REQUEST_SEC_MSG -3 X'FFFFFFFD'
MQXCC_SUPPRESS_FUNCTION -1 X'FFFFFFFE'
MQXCC_OK 0 X'00000000'

MQXPT_* (Transmission protocol type)

See the TransportType field described in “MQCD – Channel definition” on page 626.

MQXPT_LU62 1 X'00000001'
MQXPT_TCP 2 X'00000002'
MQXPT_NETBIOS 3 X'00000003'
MQXPT_SPX 4 X'00000004'
MQXPT_DECNET 5 X'00000005'
MQXPT_UDP 6 X'00000006'
Constants

MQXR_* (Exit reason)

See the ExitReason field described in "MQCXP – Channel exit parameter" on page 666.

MQXR_INIT 11 X'0000000B'
MQXR_TERM 12 X'0000000C'
MQXR_MSG 13 X'0000000D'
MQXR_XMIT 14 X'0000000E'
MQXR_SEC_MSG 15 X'0000000F'
MQXR_INIT_SEC 16 X'00000010'
MQXR_RETRY 17 X'00000011'
MQXR_AUTO_CLUSSDR 18 X'00000012'
MQXR_AUTO_RECEIVER 19 X'00000013'
MQXR_END BATCH 25 X'00000019'
MQXR_ACK_RECEIVED 26 X'0000001A'
MQXR_AUTO_SVRCONN 27 X'0000001B'
MQXR_AUTO_CLUSRCVR 28 X'0000001C'

MQXR2_* (Secondary exit response)

See the ExitResponse2 field described in "MQCXP – Channel exit parameter" on page 666.

MQXR2_PUT_WITH_DEF_ACTION 0 X'00000000'
MQXR2_USE_AGENT_BUFFER 0 X'00000000'
MQXR2_DEFAULT_CONTINUATION 0 X'00000000'
MQXR2_PUT_WITH_DEF_USERID 1 X'00000001'
MQXR2_PUT_WITH_MSG_USERID 2 X'00000002'
MQXR2_USE_EXIT_BUFFER 4 X'00000004'
MQXR2_CONTINUE_CHAIN 8 X'00000008'
MQXR2_SUPPRESS_CHAIN 16 X'00000010'

MQXT_* (Exit identifier)

See the ExitId field described in "MQCXP – Channel exit parameter" on page 666.

MQXT_CHANNEL_SEC_EXIT 11 X'0000000B'
MQXT_CHANNEL_MSG_EXIT 12 X'0000000C'
MQXT_CHANNEL_SEND_EXIT 13 X'0000000D'
MQXT_CHANNEL_RCV_EXIT 14 X'0000000E'
MQXT_CHANNEL_MSG_RETRY_EXIT 15 X'0000000F'
MQXT_CHANNEL_AUTO_DEF_EXIT 16 X'00000010'

MQXUA_* (Exit user area)

See the ExitUserArea field described in "MQCXP – Channel exit parameter" on page 666.

MQXUA_NONE X'00...00' (16 nulls)

For the C programming language, the following array version is also defined:
MQXWD_* (Exit wait descriptor structure identifier)

See the StrucId field described in "MQXWD – Exit wait descriptor" on page 685.

MQXWD_STRUC_ID

'MQXWD_STRUC_ID_ARRAY

For the C programming language, the following array version is also defined:

MQXWD_STRUC_ID_ARRAY

'MQXWD_* (Exit wait descriptor version)

See the Version field described in "MQXWD – Exit wait descriptor" on page 685.

MQXWD_VERSION_1
Constants
Appendix C. Queue name resolution

This appendix describes queue name resolution as performed by queue managers at both sending and receiving ends of a channel.

In larger networks, the use of queue managers has a number of advantages over other forms of communication. These advantages derive from the name resolution function in DQM and the main benefits are:

- Applications do not need to make routing decisions
- Applications do not need to know the network structure
- Network links are created by systems administrators
- Network structure is controlled by network planners
- Multiple channels can be used between nodes to partition traffic

Referring to Figure 149, the basic mechanism for putting messages on a remote queue, as far as the application is concerned, is the same as for putting messages on a local queue:

- The application putting the message issues MQOPEN and MQPUT calls to put messages on the target queue.
- The application getting the messages issues MQOPEN and MQGET calls to get the messages from the target queue.
Queue name resolution

If both applications are connected to the same queue manager then no inter-queue manager communication is required, and the target queue is described as local to both applications.

However, if the applications are connected to different queue managers, two MCAs and their associated network connection are involved in the transfer, as shown in the figure. In this case, the target queue is considered to be a remote queue to the putting application.

The sequence of events is as follows:
1. The putting application issues MQOPEN and MQPUT calls to put messages to the target queue.
2. During the MQOPEN call, the name resolution function detects that the target queue is not local, and decides which transmission queue is appropriate. Thereafter, on the MQPUT calls associated with the MQOPEN call, all messages are placed on this transmission queue.
3. The sending MCA gets the messages from the transmission queue and passes them to the receiving MCA at the remote computer.
4. The receiving MCA puts the messages on the target queue, or queues.
5. The getting application issues MQOPEN and MQGET calls to get the messages from the target queue.

Note: Only step 1 and step 5 involve application code; steps 2 through 4 are performed by the local queue managers and the MCA programs. The putting application is unaware of the location of the target queue, which could be in the same processor, or in another processor on another continent.

The combination of sending MCA, the network connection, and the receiving MCA, is called a message channel, and is inherently a unidirectional device. Normally, it is necessary to move messages in both directions, and two channels are set up for this, one in each direction.
Queue name resolution is vital to DQM. It removes the need for applications to be concerned with the physical location of queues, and insulates them against the details of networks. A systems administrator can move queues from one queue manager to another, and change the routing between queue managers without applications needing to know anything about it.

In order to uncouple from the application design the exact path over which the data travels, it is necessary to introduce a level of indirection between the name used by the application when it refers to the target queue, and the naming of the channel over which the flow occurs. This indirection is achieved using the queue name resolution mechanism.

In essence, when an application refers to a queue name, the name is mapped by the resolution mechanism either to a transmission queue or to a local queue that is not a transmission queue. In the case of mapping to a transmission queue, a second name resolution is needed at the destination, and the received message is placed on the target queue as intended by the application designer. The application remains unaware of the transmission queue and channel used for moving the message.

**Note:** The definition of the queue and channel is a system management responsibility and can be changed by an operator or a system management utility, without the need to change applications.

An important requirement for the system management of message flows is that alternative paths should be provided between queue managers. For example, business requirements might dictate that different classes of service should be sent over different channels to the same destination. This is a system management decision and the queue name resolution mechanism provides a very flexible way to achieve it. The next section describes in detail how this is done, but the basic idea is to use queue name resolution at the sending queue manager to map the queue name supplied by the application to the appropriate transmission queue for the type of traffic involved. Similarly at the receiving end, queue name resolution maps the name in the message descriptor to a local (not a transmission) queue or again to an appropriate transmission queue.

Not only is it possible for the forward path from one queue manager to another to be partitioned into different types of traffic, but the return message that is sent to the reply-to queue definition in the outbound message can also use the same traffic partitioning. Queue name resolution satisfies this requirement and the application designer need not be involved in these traffic partitioning decisions.

The point that the mapping is carried out at both the sending and receiving queue managers is an important aspect of the way name resolution works. This allows the queue name supplied by the putting application to be mapped to a local queue or a transmission queue at the sending queue manager, and again remapped to a local queue or a transmission queue at the receiving queue manager.

Reply messages from receiving applications or MCAs have the name resolution carried out in exactly the same way, allowing return routing over specific paths by means of queue definitions at all the queue managers on route.
Queue name resolution

**How queue name resolution works**

The [WebSphere MQ Application Programming Guide](#) provides the rules for queue name resolution.
Appendix D. Configuration file stanzas for distributed queuing

This appendix shows the stanzas in the queue manager configuration file that relate to distributed queuing. It applies to:

- The queue manager configuration file for MQSeries for OS/2 Warp, called qm.ini
- The queue manager configuration file for WebSphere MQ on UNIX systems, called qm.ini
- The queue manager initialization file for WebSphere MQ for iSeries, called qm.ini.

Notes:

1. The stanzas in the QMINI file for Compaq NonStop Kernel are different and are described in the MQSeries for Compaq NonStop Kernel System Administration manual.

2. WebSphere MQ for Windows uses the registry. Use the WebSphere MQ Services snap-in within the Microsoft Management Console (MMC) to make equivalent changes to the configuration information.

The stanzas that relate to distributed queuing are:

- CHANNELS
- TCP
- LU62
- NETBIOS
- SPX
- EXITPATH
- UDP (WebSphere MQ for AIX only)
- Transport (WebSphere MQ for AIX only)

Figure 150 on page 716 shows the values that you can set using these stanzas. When you are defining one of these stanzas, you do not need to start each item on a new line. You can use either a semicolon (;) or a hash character (#) to indicate a comment.
Configuration file stanzas

CHANNELS:
MAXCHANNELS=n ; Maximum number of channels allowed, the
default value is 100
MAXACTIVECHANNELS=n ; Maximum number of channels allowed to be active at
any time, the default is the value of MaxChannels
MAXINITIATORS=n ; Maximum number of initiators allowed, the
default value is 3 (see note 1)
MQIBINDTYPE=type ; Whether the binding for applications is to be
"fastpath" or "standard".
The default is "standard". (see note 2)
ADOPTNEWMCA=chlttype ; Stops previous process if channel fails to start.
The default is "NO".
ADOPTNEWMCATIMEOUT=n ; Specifies the amount of time that the new
process should wait for the old process to end.
The default is 60.
ADOPTNEWMCACHE= typecheck ; Specifies the type checking required.
For FAP1, FAP2, and FAP3, "NAME" and
"ADDRESS" is the default.
For FAP4 and later, "NAME",
"ADDRESS", and "QM" is the
default.
TCP: ; TCP entries
PORT=n ; Port number, the default is 1414
LIBRARY1=DLLName1 ; Name of TCP Sockets DLL (OS/2 only)
LIBRARY2=DLLName2 ; Same as above if code is in two libraries (OS/2 only)
KEEPALIVE=Yes ; Switch TCP/IP KeepAlive on
IPADDR=Addr/Name ; TCP/IP address or name for Listener
LU62: ; LU 6.2 entries (OS/2 only)
TPNAME=name ; TP Name to start on remote side
LIBRARY1=DLLName1 ; Name of APPC DLL (see note 3)
LIBRARY2=DLLName2 ; Same as above if code is in two libraries (see note 3)
LOCALU=name ; LU to use on local system (OS/2 only)
NETBIOS: ; NetBIOS entries (OS/2 only)
LOCALNAME=name ; The name this machine will be known as on the LAN
ADAPTERNUM=n ; LAN adapter number, the default is adapter 0
NUMSESS=n ; Number of sessions to allocate, the default is 1
NUMCMDS=n ; Number of commands to allocate, the default is 1
NUMNAMES=n ; Number of names to allocate, the default is 1
LIBRARY1=DLLName1 ; Name of NetBIOS DLL
LIBRARY2=DLLName2 ; Same as above if code is in two libraries (OS/2 only)
SPX: ; SPX entries (OS/2 only)
SOCKET=n ; The socket number, the default is 5EB6
BOARDNUM=0 ; LAN adapter number, the default is adapter 0 (OS/2 only)
KEEPALIVE=Yes ; Switch on "watchdog" to monitor sessions (OS/2 only)
LIBRARY1=DLLName1 ; Name of SPX DLL
LIBRARY2=DLLName2 ; Same as above if code is in two libraries (OS/2 only)
EXITPATH: ; Location of user exits (MQSeries for AIX,
HP-UX, OS/2 Warp, and Solaris only)
EXITPATHS= ; String of directory paths
UDP: ; UDP entries (AIX only)
ACKREQ_TIMEOUT=n ; request for acknowledgement timeout
attribute specifies the time, in
seconds, that the internal state
machines will wait for a protocol
datagram before assuming that the
datagram has been lost and retrying.
The default is 5.
ACKREQ_RETRY=n ; request for acknowledgment retry
attribute specifies the number of times
that the internal state machines will
resend protocol datagrams before giving
up completely and causing a channel to
close. (All the counts are reset to
zero after success and thus are not
cumulative.) The default is 60.
CONNECT_TIMEOUT=n ; connect request timeout attribute
specifies the time, in seconds, that
the internal state machines will wait
for a protocol datagram before assuming
Notes:

1. MAXINITIATORS applies only to WebSphere MQ for AIX, WebSphere MQ for iSeries, WebSphere MQ for HP-UX, MQSeries for OS/2 Warp, and WebSphere MQ for Solaris.

2. MQIBINDTYPE applies only to WebSphere MQ for AIX, WebSphere MQ for iSeries, WebSphere MQ for HP-UX, MQSeries for OS/2 Warp, and WebSphere MQ for Solaris.

3. The default values for LIBRARY1 and LIBRARY2 are as follows:
   - **TCP**
     - SO32DLL and TCP32DLL (OS/2)
   - **LU 6.2**
     - APPC and ACSSVC (OS/2)
   - **NetBIOS**
     - ACSNETB (OS/2)
   - **SPX**
     - IPXCALLS.DLL and SPXCALLS.DLL (OS/2)

For more information about the qm.ini file and the other stanzas in it, refer to the WebSphere MQ System Administration Guide book for WebSphere MQ for AIX, HP-UX, Solaris, and Windows, and MQSeries V5.1 for Compaq Tru64 UNIX, and OS/2 Warp, and to the WebSphere MQ for iSeries V5.3 System Administration book for WebSphere MQ for iSeries.
Further intercommunication considerations
Appendix E. Notices

This information was developed for products and services offered in the United States. IBM may not offer the products, services, or features discussed in this information in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user’s responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this information. The furnishing of this information does not give you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

IBM World Trade Asia Corporation
Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106, Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law:

INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION “AS IS” WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY, OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the information. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this information at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.
Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM United Kingdom Laboratories,
Mail Point 151,
Hursley Park,
Winchester,
Hampshire,
England
SO21 2JN.

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this information and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Programming License Agreement, or any equivalent agreement between us.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. You may copy, modify, and distribute these sample programs in any form without payment to IBM for the purposes of developing, using, marketing, or distributing application programs conforming to IBM’s application programming interfaces.
## Trademarks

The following terms are trademarks of International Business Machines Corporation in the United States, or other countries, or both:

<table>
<thead>
<tr>
<th>Term</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACF/VTAM</td>
<td>Advanced Peer-to-Peer Networking</td>
</tr>
<tr>
<td>APPN</td>
<td>AS/400</td>
</tr>
<tr>
<td>CICS</td>
<td>CICS/VSE</td>
</tr>
<tr>
<td>DB2</td>
<td>FFST First Failure Support Technology</td>
</tr>
<tr>
<td>IBM</td>
<td>IBMLink Integrated Language Environment</td>
</tr>
<tr>
<td>iSeries</td>
<td>Language Environment MQSeries</td>
</tr>
<tr>
<td>MVS/ESA</td>
<td>OpenEdition OS/2</td>
</tr>
<tr>
<td>OS/390</td>
<td>OS/400 RACF</td>
</tr>
<tr>
<td>RPG/400</td>
<td>S/390 SP2</td>
</tr>
<tr>
<td>System/390</td>
<td>VisualAge VSE/ESA</td>
</tr>
<tr>
<td>VTAM</td>
<td>WebSphere z/OS</td>
</tr>
</tbody>
</table>

Lotus and LotusScript are trademarks of Lotus Development Corporation in the United States, or other countries, or both.

Intel and all Intel-based trademarks and logos are trademarks or registered trademarks of Intel Corporation in the United States and/or other countries.

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States and/or other countries.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States and/or other countries.

UNIX is a registered trademark in the United States and/or other countries licensed exclusively through X/Open Company Limited.

Other company, product, and service names may be trademarks or service marks of others.
Further intercommunication considerations
Index

A
active channels, maximum number 64
add routing entry 517
addressing information 25
addrtge 517
administration, channel 59
Adopting an MCA 67
Adopting an MCA, information on 691
agent
intra-group queuing 463
AgentBuffer parameter 615
AgentBufferLength parameter 615
aliases 25
ALTDATE attribute 80
alter channel
z/OS 345
z/OS using CICS 388
Alter option 402
alternate channels 15
ALTTIME attribute 80
AMQCRCTA channel program 507
AMQCRS6A channel program 123
AMQCRSTA channel program 123, 507
AMQRMCLA channel program 507
APC pathway definition, example 313
APPC/MVS, defining a connection 364, 446
applications, trusted 12, 126
ARM (Automatic Restart Management) 360
assured delivery 22
AT&T GIS SNA Server 246
Attachmate PathWay 298
attributes
ALTDATE 80
alter date 80
alter time 80
ALTTIME 80
auto start 80
AUTOSTART 80
back out in-doubt messages
Compaq NonStop Kernel 119
Compaq OpenVMS Alpha 119
OS/2 119
OS/400 501
UNIX systems 119
Windows systems 119
balanced
Workload 442
back out in-doubt messages
batch heartbeat 80
batch interval 81
batch size 81
BATCHHB 80
BATCHINT 81
BATCHSZ 81
CHANNEL 82
channel description 86
channel name 82
channel type 83
CLIENTYPE 83
CLUSNL 83
CLUSTER 83
cluster name 83
cluster namelist 83
communication connection identifier 84
CONNAME 84
connection name 84
CONVERT 85
convert message 85
attributes (continued)
DESCR 86
DISCNT 86
disconnect interval 86
HBINT 87
heartbeat interval 87
KAINT 87
Keepalive interval 87
local address 88
LOCALADDR 88
long retry count 88
long retry interval 89
LONGRTY 88
LONGTMR 89
LU 6.2 mode name 89
LU 6.2 TP name 89
maximum message length 90
maximum transmission size 90
MAXMSGL 90
MCA name 91
MCA type 91
MCA user 91
MCANAME 91
MCATYPE 91
MCUSER 91
message exit name 92
message exit user data 92
message retry count 92
message retry interval 93
message-retry exit name 92
message-retry exit user data 92
mode name 89
MODENAME 89
MRDATA 92
MREXIT 92
MRRTY 92
MRRTMR 93
MSQDATA 92
MSGEXIT 92
NETPRTY 93
network-connection priority 93
nonpersistent message speed 93
NPMSPEED 93
password 93
profile name, CICS 83
PUT authority 94
PUTAUT 94
QMNAME 95
queue manager name 95
RCVDATE 95
RCVEXIT 95
receive exit name 95
receive exit user data 95
SCYDATA 96
SCYEXIT 96
security exit name 96
security exit user data 96
send exit name 96
send exit user data 96
SEND DATA 96
SEND EXIT 96
attributes (continued)
sequence number wrap 96
sequential delivery 97
SEQWRAP 96
short retry count 97
short retry interval 97
SHORTRTY 97
SHORTTMR 97
SSL Cipher Specification 98
SSL Client Authentication 98
SSL Peer 98
SSLCAUTH 98
SSLCIPH 98
SSLPEER 98
target system identifier 99
TPNAME 89
transaction identifier 99
transmission protocol 99
transmission queue name 99
transport type 99
TRPTYPE 99
user ID 100
USERID 100
XMITQ 99
authority, PUT 94
auto-definition exit program 583
auto-definition of channels 60
automatic channel reconnect for TCP/IP 360
Automatic Restart Management (ARM) 360
AUTOSTART 80
bind type 126
BINDING channel state 62
binding, fastpath 126
browsing a channel 388, 492
channel (continued)
definition file
  Compaq NonStop Kernel 110
  Compaq OpenVMS Alpha 110
  OS/2 110
  OS/400 485
  UNIX systems 110
  Windows systems 110
  z/OS using CICS 383
definition, what is it? 57
delete 116, 499
  z/OS 345
  z/OS using CICS 402
description 86
display
  Compaq NonStop Kernel 116
  Compaq OpenVMS Alpha 116
  OS/2 116
  OS/400 499
  UNIX systems 116
  Windows systems 116
  z/OS using CICS 397
display status
  Compaq NonStop Kernel 116
  Compaq OpenVMS Alpha 116
  OS/2 116
  OS/400 using CICS 396
  OS/400 499
  UNIX systems 116
  Windows systems 116
display, z/OS 345
displaying 114, 499
displaying settings
  Compaq NonStop Kernel 116
  Compaq OpenVMS Alpha 116
  OS/2 116
  OS/400 using CICS 396
displaying status 499
  Compaq NonStop Kernel 116
  Compaq OpenVMS Alpha 116
  OS/2 116
  OS/400 499
  UNIX systems 116
  Windows systems 116
  z/OS using CICS 396
displaying status 499
  Compaq NonStop Kernel 116
  Compaq OpenVMS Alpha 116
  OS/2 116
  OS/400 499
  UNIX systems 116
  Windows systems 116
  z/OS using CICS 396

detecting 61
end 500
error 65
  restarting after 69
exit current function 399
fastpath binding 126
  find 402
in doubt 70
in-doubt channels 70
initial data negotiation 61
initiator
  AIX, OS/2, HP-UX, Solaris, and
  Windows systems 122, 123
overview 10
  starting 122
  stopping 123
  z/OS 346

channel (continued)
listener
  overview 10
  start, OS/400 499
  start, z/OS 348
  stop, z/OS 349
  STOPMQMLSR command 499
  trusted 12
menu-bar choice 405
monitoring 60
MQI 8
OS/400 501
resolve 501
ping
  Compaq NonStop Kernel 116
  Compaq OpenVMS Alpha 116
  OS/2 116
  OS/400 499
  UNIX systems 116
  Windows systems 116
  z/OS 351
  z/OS using CICS 398
planning form 699
preparing 60
program types
  Compaq NonStop Kernel 123
  Compaq OpenVMS Alpha 123
  OS/2 123
  UNIX systems 123
  WebSphere MQ for iSeries 507
  Windows systems 123
programs 507
  AMQCCCLA 507
  AMQCRCTA 507
  AMQCRS6A 123, 507
  AMQRSTA 123
  AMQRMCLA 507
  z/OS using CICS 383
pull-down menu 405
queries 67
receiver 8
receiving parameters 59
refuses to run 689
renaming
  Compaq NonStop Kernel 115
  Compaq OpenVMS Alpha 115
  OS/2 115
  OS/400 497
  UNIX Systems 115
  Windows systems 115
  z/OS using CICS 389
requester 8
requester-sender 9
requester-server 8
reset
  z/OS 351
  z/OS using CICS 394
reset
  Compaq NonStop Kernel 119
  Compaq OpenVMS Alpha 119
  OS/2 119
  OS/400 501
  UNIX systems 119
  Windows systems 119
resolving
  Compaq NonStop Kernel 119
  Compaq OpenVMS Alpha 119

channel (continued)
definition
  MQ_CHANNEL_AUTO_DEF_EXIT 619
  MQ_CHANNEL_EXIT 614
  MQ_TRANSPORT_EXIT 622
  MQXWAIT 624
calls
detailed description
  CapabilityFlags field 675
  CEDA CICS transaction 418
  change definition, channel 116, 498
  Change option 498
  channel administration 59
  altering 388
  attributes 115
  auto-definition 60
auto-definition exit program 583
browsing 388, 492
change definition 116, 498
channel control function
  Compaq NonStop Kernel 109
  Compaq OpenVMS Alpha 109
  OS/2 109
  OS/400 485
  UNIX systems 109
  Windows systems 109
characteristics
  Compaq NonStop Kernel 123
  Compaq OpenVMS Alpha 123
  OS/2 123
  OS/400 507
  UNIX systems 123
  Windows systems 123
  z/OS using CICS 383
client-connection 8
cluster-receiver 10
cluster-sender 9
command queue
  z/OS 358
  configuration 554
control commands 60
copy definition 399, 498
create definition
  Compaq NonStop Kernel 115
  Compaq OpenVMS Alpha 115
  OS/2 115
  OS/400 497
  UNIX Systems 115
  Windows systems 115
  z/OS using CICS 400
creating 113, 387, 488
creating your own defaults 401, 497
default values supplied by OS/390
using CICS 401, 404
default values supplied by WebSphere
MQ for iSeries 497
define
  z/OS 344
  z/OS using CICS 404

channel
  caller
    MCA 10
caller, responder 10
calls
detailed description
  MQ_CHANNEL_AUTO_DEF_EXIT 619
  MQ_CHANNEL_EXIT 614
  MQ_TRANSPORT_EXIT 622
  MQXWAIT 624
CapabilityFlags field 675
CEDA CICS transaction 418
change definition, channel 116, 498
Change option 498
channel administration 59
altering 388
attributes 115
auto-definition 60
auto-definition exit program 583
browsing 388, 492
change definition 116, 498
channel control function
  Compaq NonStop Kernel 109
  Compaq OpenVMS Alpha 109
  OS/2 109
  OS/400 485
  UNIX systems 109
  Windows systems 109
characteristics
  Compaq NonStop Kernel 123
  Compaq OpenVMS Alpha 123
  OS/2 123
  OS/400 507
  UNIX systems 123
  Windows systems 123
  z/OS using CICS 383
client-connection 8
cluster-receiver 10
cluster-sender 9
command queue
  z/OS 358
  configuration 554
control commands 60
copy definition 399, 498
create definition
  Compaq NonStop Kernel 115
  Compaq OpenVMS Alpha 115
  OS/2 115
  OS/400 497
  UNIX Systems 115
  Windows systems 115
  z/OS using CICS 400
creating 113, 387, 488
creating your own defaults 401, 497
default values supplied by OS/390
using CICS 401, 404
default values supplied by WebSphere
MQ for iSeries 497
define
  z/OS 344
  z/OS using CICS 404
channel (continued)
resolving (continued)
OS/2 119
OS/400 501
UNIX Systems 119
Windows systems 119
z/OS 352
z/OS using CICS 395
restart 61
restarting when stopped 69
resync, z/OS using CICS 393
run 115, 490
segregating messages 15
selecting 492
selecting z/OS using CICS 386
sender-receiver 8
sequence numbers 59
server-connection 8
server-receiver 9
sharing 14
start 61
Compaq NonStop Kernel 115, 117
Compaq OpenVMS Alpha 115, 117
OS/2/115, 117
OS/400 490, 499
UNIX systems 117
UNIX Systems 115
Windows systems 115, 117
z/OS 349
z/OS using CICS 389
startup negotiation errors 689
startup, data negotiation 61, 572, 574
state 61, 62
status 59
stopping 67, 500
Compaq NonStop Kernel 118
Compaq OpenVMS Alpha 118
OS/2/118
OS/400 500
UNIX systems 118
Windows systems 118
z/OS 352
z/OS using CICS 391, 415
switching 693
synchronizing 393, 572
test, z/OS 351
transport-retry exit program 584
triggering 20, 390
Compaq NonStop Kernel 121
OS/2 121
UNIX systems 121
Windows systems 121
z/OS 357
z/OS using CICS 390
trusted 126
types 83, 115, 123, 384
using alternate channels 15
working with z/OS using CICS 386
CHANNEL attribute 82
channel attributes 601
channel auto-definition exit, introduction 12
channel configuration (continued)
MQSeries for AT&T GIS UNIX 252
MQSeries for Compaq Tru64 UNIX 216
channel control error messages 687
channel control function 59
Compaq NonStop Kernel 109
Compaq OpenVMS Alpha 109
OS/2 109
OS/400 485
UNIX systems 109
Windows systems 109
z/OS 341
z/OS using CICS 383
channel definition file
Compaq NonStop Kernel 110
Compaq OpenVMS Alpha 110
OS/2/110
OS/400 485
UNIX systems 110
Windows systems 110
z/OS using CICS 383
channel description 86
channel exit
compatible receive exit 581
considerations for pipelining 189
reserving space in send exit 581
channel exits
auto-definition 583
message 581
message-retry 583
receive 578
security 574
send 578
transport-retry 584
channel functions
Compaq NonStop Kernel 115
Compaq OpenVMS Alpha 115
OS/2 115
UNIX systems 115
Windows systems 115
channel initiator
display, z/OS 346
overview 10
retries 65, 89
runmqchi command, Compaq NonStop Kernel 117
runmqchi command, MQSeries for OS/2 Warp 117
runmqchi command, WebSphere MQ for Windows 117
running the MCA as a thread 91
start, OS/2, Windows systems, Compaq OpenVMS Alpha, Compaq NonStop Kernel, and UNIX systems 122
start, OS/400 500
start, z/OS 346
channel listeners
overview 10
start, OS/400 500
start, z/OS 348
stop, z/OS 349
STRMQMCHLI command 500
channel listener
overview 10
start, OS/400 500
start, z/OS 348
stop, z/OS 349
STRMQMLSR command 500
trusted 12
channel name attribute 82
channel planning example
Compaq OpenVMS Alpha 323
OS/2 323
OS/400 541
UNIX systems 323
Windows NT 323
channel planning examples
z/OS 377, 457
channel planning form, how to use 699
channel programs
Compaq NonStop Kernel 123
Compaq OpenVMS Alpha 123
OS/2 123
UNIX systems 123
WebSphere MQ for iSeries 507
Windows systems 123
z/OS using CICS 383
channel refuses to run 689
channel settings panel, OS/390 using CICS 406
channel startup negotiation errors 689
channel states
BINDING 61
INACTIVE 65
OS/400 508
channel status
display, Compaq NonStop Kernel 116
display, Compaq OpenVMS Alpha 116
display, OS/2 116
display, OS/390 using CICS 396
display, OS/400 499
display, UNIX systems 116
display, Windows systems 116
display, z/OS 354
channel type attribute 83
channel-exit program 610
channel-exit programs 571, 610
channel definition structure, MQCD 585
channel-definition structure, MQCD 585
data buffer 585
introduction 12
MQSeries for AT&T GIS UNIX 598
MQSeries for Compaq NonStop Kernel 600
MQSeries for Compaq OpenVMS Alpha 595
MQSeries for Compaq OpenVMS UNIX 596
MQSeries for OS/2 Warp 589
channel-exit programs (continued)
MQSeries for SINIX and DC/OSx 600
parameter structure, MQCXP 585
supplied programs, DCE 604
WebSphere MQ for AIX 593
WebSphere MQ for HP-UX 597
WebSphere MQ for iSeries 589
WebSphere MQ for Solaris 599
WebSphere MQ for Windows 591
WebSphere MQ for z/OS using CICS 588
WebSphere MQ for z/OS without CICS 587
Windows 3.1 client 591
Windows 95 and Windows 98 client 591
Windows NT client 591
writing and compiling 585
channel, performance improvement 189
ChannelDefinition parameter
MQ_CHANNEL_AUTO_DEF_EXIT call 619
MQ_CHANNEL_EXIT call 614
ChannelExitParms parameter
MQ_CHANNEL_AUTO_DEF_EXIT call 619
MQ_CHANNEL_EXIT call 614
ChannelName field 629
CHANNELS stanza 715
channels, alternate to 15
ChannelType field 630
CHLTYPE attribute 83
CICS
CEDA INSTALL command 419
installing communication connection 419
regions 384
transaction
CEDA 418
CKMC 384
CKSG 413
Cisco MultiNet for Compaq OpenVMS Alpha 297
CKMC CICS transaction 384
CKSG CICS transaction 413
class of service 439
class of routing entry 519
class of service 47
class, problem determination 694
CLUSNL attribute 83
CLUSTER attribute 83
cluster channels, z/OS 355
cluster components 6
cluster name attribute 83
cluster namelist attribute 83
cluster-receiver 10
cluster-receiver channel 8
cluster-sender 9
cluster-sender channel 8
clustering
with intra-group queuing 468
ClusterPtr field 648
clusters
choosing transmission queue 39
components 6
concentrating messages 44
distribution lists 46
message flow 35
networking considerations 52
passing messages 41
putting messages 38
queue sharing with 442
reply-to queue 47
return routing 53
separating message flows 42
using 16
Clusters and queue-sharing groups 442
ClustersDefined field 648
command queue channel, z/OS 358
command validation 71
commit in-doubt messages
Compaq NonStop Kernel 119
Compaq OpenVMS Alpha 119
OS/2 119
OS/400 501
UNIX systems 119
Windows systems 119
committed messages
Compaq NonStop Kernel 119
Compaq OpenVMS Alpha 119
OS/2 119
OS/400 501
UNIX systems 119
Windows systems 119
communication
between CICS systems attached to one queue manager 419
between queue managers 417
intersystem (ISC) 417
communications examples
ICE 317
SNAX 309
TCP/IP 321
Communications Manager/2 134
Communications Manager/2 133, 134
Communications Server for AIX V5 203
Communications Server for Windows NT 172
communications setup, Compaq NonStop Kernel 309
communications side object
OS/400 513, 514
z/OS 364, 446
CompCode parameter 624
components of
shared queuing 439
components of distributed-queuing environment 13
channel initiator 10
channel listener 10
message channel 8
message channel agent 10
transmission queue 10
Components of shared queuing 439
components, cluster 6
compression of data 578
concentrating messages 44
concentrators 31
concepts
shared queuing 439
concepts of
intra-group queuing 461
concepts of intercommunication 3, 16, 22
Concepts of queue-sharing groups 439
configuration
MQSeries for AT&T GIS UNIX 252
MQSeries for Compaq Tru64 UNIX 215
MQSeries for OS/2 Warp 159
MQSeries for VSE/ESA 554
WebSphere MQ for AIX 209
WebSphere MQ for HP-UX 238
WebSphere MQ for iSeries 534
WebSphere MQ for Solaris 289
WebSphere MQ for Windows 183
WebSphere MQ for z/OS 371, 426, 454
configuration file 73
Compaq NonStop Kernel 74
Compaq OpenVMS Alpha 74
OS/2 74
UNIX systems 74
configuration worksheet 549
configurations
intra-group queuing 466
configuring the UDP transport-retry exit 385
CONNAME attribute 84
connection
APPC/MVS, z/OS 361, 445
deciding upon
z/OS 361, 445
DECnet Phase IV 304
DECnet Phase V 304
defining APPC/MVS (LU 6.2) 364, 446
defining LU 6.2
Compaq OpenVMS Alpha 299
OS/2 132
OS/400 513
UNIX systems 194
Windows systems 132
installing 419
LU 6.2
Compaq NonStop Kernel 307
Compaq OpenVMS Alpha 295
OS/2 129
OS/400 511
UNIX systems 191
Windows NT 129
z/OS 361, 445
z/OS using CICS 418
NetBIOS
OS/2 129
Windows NT 129
SPX
OS/2 129
Windows NT 129
switching 694
connection (continued)
TCP
  Compaq NonStop Kernel 307
  Compaq OpenVMS Alpha 295
  OS/2 129
  OS/400 511
  UNIX systems 191
  Windows NT 129
  z/OS 361, 445
UDP
  UNIX systems 191
connection name 84
  for function shipping 419
ConnectionName field 639
  constants, values of 703
  channel capability flags
    (MQCF_*) 704
  channel data conversion
    (MQCDC_*) 704
  channel definition structure length
    (MQCD_*) 704
  channel definition structure version
    (MQCD_*) 704
  channel type (MQCHT_*) 704
  channel-exit parameter structure identifier
    (MQCXP_*) 705
  channel-exit parameter structure version
    (MQCXP_*) 705
  completion codes (MQCC_*) 704
  keepalive interval (MQKAI_*) 705
  lengths of character string and byte fields
    (MQ_*) 703
  MCA type (MQMCAT_*) 705
  nonpersistent message speed
    (MQNPMS_*) 706
  put authority (MQPA_*) 706
  reason codes (MQRC_*) 706
  secondary exit response
    (MQXR2_*) 708
  security identifier (MQSID_*) 706
  security identifier type
    (MQSIDT_*) 707
  SSL client authentication
    (MQSCA_*) 706
  transmission protocol type
    (MQXPT_*) 707
  transport retry exit structure identifier
    (MQTXP_*) 707
  transport retry exit structure version
    (MQTXP_*) 707
context security 94
  control commands, channel 60
  conversion failure, problem determination 691
  conversion of data 59
  CONVERT attribute 85
  convert message 85
  coordination with adjacent systems 44

D
data
  compression 578
  conversion 581
  decompression 578
  encryption 581
  negotiation 20, 60
  data conversion 75
  data types, detailed description
    MQCD 626
    MQCXP 666
    MQTXP 681
    MQXWD 685
  DataConversion field 636
  Datatid field 683
  DataLength field 683
  DataLength parameter 614
  DCE
    supplied exit programs 604
  DDNS
    registration time for 691
dead-letter queue 52
  Compaq NonStop Kernel 124
  Compaq OpenVMS Alpha 124
  OS/2 124
  overview 13
  problem determination 688
  processing 688
  UNIX systems 124
  WebSphere MQ for iSeries 509
  Windows systems 124
  z/OS 124
  DECF
  DECF Phases I-IV 295
  DECF Phase IV connection 304
  DECF Phase V connection 304
  decompression of data 578
  default channel values
    OS/400 497
    z/OS using CICS 401, 404
default object creation 113
define channel
  z/OS 344
defining
  an LU 6.2 connection
    Compaq OpenVMS Alpha 299
    OS/2 132
    OS/400 513
    UNIX systems 194
  Windows systems 132
  APPC/MVS (LU 6.2) connection
    z/OS 364, 446
  objects
    z/OS 357
  queues
    z/OS 344
  z/OS using CICS 404
definition file
  Compaq NonStop Kernel 110
  Compaq OpenVMS Alpha 110
  OS/2 110
  UNIX systems 485
  Windows systems 110
  z/OS using CICS 383
delete channel
  distributed platforms 116
  OS/400 499
  z/OS 345
  z/OS using CICS 402
delivery, messages 22
  Desc field 631
  DESC attribute 86
description, channel 86
destAddress parameter 622
destAddressLength parameter 622
destination queue 42
dial-up support 691
digital TCP/IP services for
  OpenVMS 296
disabled receiver channels 117, 499
disaster recovery 693
  DISCINT attribute 86
  DisInterval field 633
disconnect interval 86
display
  option 499
  settings 397
  status 396
  z/OS, DQM 346
display channel
  Compaq NonStop Kernel 114
  Compaq OpenVMS Alpha 114
  OS/2 114
  OS/400 499
  UNIX systems 114
  Windows systems 114
  z/OS 345
display channel initiator
  z/OS 346
display channel status
  z/OS 354
display channel status
  Compaq NonStop Kernel 114
  Compaq OpenVMS Alpha 114
  OS/2 114
  UNIX systems 114
Display channel status (continued)
Windows systems 114
display DQM 346
display settings 397
display status 396
distributed queuing
components 7, 13
functions 57
with intra-group queuing 466
distributed queuing in z/OS using
CICS 413
distributed queuing with queue-sharing
groups 439
distributed-queuing management in
WebSphere MQ for iSeries 503
distribution lists 46, 59
diverting message flows 45
DQM
display, z/OS 346
DQM panels
z/OS using CICS 384

E
ECB field 686
edit
alter, OS/390 using CICS 402
change
Compaq NonStop Kernel 116
Compaq OpenVMS Alpha 116
OS/2 116
OS/400 498
UNIX systems 116
Windows systems 116
copy
OS/400 498
z/OS using CICS 399
create
Compaq NonStop Kernel 115
Compaq OpenVMS Alpha 115
OS/2 115
OS/400 497
UNIX systems 115
Windows systems 115
z/OS using CICS 400
delete
Compaq NonStop Kernel 116
Compaq OpenVMS Alpha 116
OS/2 116
OS/400 499
UNIX systems 116
Windows systems 116
z/OS using CICS 402
find
z/OS using CICS 402
menu-bar choice
OS/390 using CICS 399
pull-down menu 399
enabling a channel to transmit
messages 61
encryption
in send exit 581
encryption of messages 571
end 118
End option 500
ending a channel 118, 500
ending SNA Listener process 302
ENMDQLSR command 123
error
at remote sites 687
channel 65
logs 118, 694
message from channel control 687
recovery 687
example
channel planning
Compaq OpenVMS Alpha 323
OS/2 323
OS/400 541
UNIX systems 323
Windows NT 323
communications setup
Compaq NonStop Kernel 309
configuration file, SINIX and
DC/OSx 329
constructions 101, 102
establishing a session using
SNAP-IX 276
flow control 35
intra-group queuing 475
local queue definition
Compaq NonStop Kernel 326
Compaq OpenVMS Alpha 326
OS/2 326
OS/400 544
UNIX systems 326
Windows NT 326
z/OS 380
process definition
Compaq NonStop Kernel 325, 326
Compaq OpenVMS Alpha 323,
326
OS/2 325, 326
UNIX systems 325, 326
Windows NT 325, 326
remote queue definition
Compaq NonStop Kernel 325, 327
Compaq OpenVMS Alpha 325,
327
OS/2 325, 327
OS/400 544, 545
UNIX systems 325, 327
Windows NT 325, 327
z/OS 379, 380
receiver channel definition
Compaq NonStop Kernel 325, 327
Compaq OpenVMS Alpha 325,
327
OS/2 325, 327
OS/400 544, 545
UNIX systems 325, 327
Windows NT 325, 327
z/OS 379
reply-to queue definition
Compaq NonStop Kernel 325
Compaq OpenVMS Alpha 325
OS/2 325
OS/400 542
UNIX systems 325
Windows NT 325
z/OS 378
running
Compaq NonStop Kernel 327
Compaq OpenVMS Alpha 327
OS/2 327
example (continued)
running (continued)
OS/400 545
UNIX systems 327
Windows NT 327
z/OS 380
send exits reserving space 581
sender channel definition
Compaq NonStop Kernel 325, 327
Compaq OpenVMS Alpha 325,
327
OS/2 325, 327
OS/400 543, 545
UNIX systems 325, 327
Windows NT 325, 327
z/OS 379, 380
transmission queue definition
Compaq NonStop Kernel 325, 326
Compaq OpenVMS Alpha 325,
326
OS/2 325, 326
OS/400 543, 545
UNIX systems 325, 326
Windows NT 325, 326
z/OS 378, 380
example configurations
MQSeries for AT&T GIS UNIX 243,
257
MQSeries for Compaq Tru64 UNIX
215, 219
MQSeries for OS/2 Warp 143, 165
WebSphere MQ for AIX 197, 214
WebSphere MQ for Compaq
OpenVMS Alpha 295, 307
WebSphere MQ for HP-UX 219, 243
WebSphere MQ for iSeries 521, 540
WebSphere MQ for Solaris 257, 295
WebSphere MQ for Windows 167, 189
WebSphere MQ for z/OS 365, 421,
447
examples
alias walk-through 51
channel initiators 11
channel listeners 11
channel names 30
channel planning
for distributed platforms 323
for OS/400 541
for z/OS 377, 457
for z/OS using CICS 431
z/OS 377, 457
choosing the transmission queue 39
cluster of queue managers 6
communication in WebSphere MQ for
iSeries, TCP connection 511
concentrating messages 44
configuration file on bight 330
configuration file on forties 331
configuration files
for Pyramid DC/OEsx 332
SINIX and DC/OEsx 329
create channel 114
creating reply-to aliases 36
defining channels 18
defining queues 19
defining remote queue definitions 36
examples (continued)
display channel 114
display channel status 114
diverting message flows 45
message channels
cluster-receiver 10
cluster-sender 9
requester-sender 9
requester-server 8
sender-receiver 8
MQSeries for AT&T GIS UNIX
configuration 243
MQSeries for OS/2 Warp
configuration 143
MQSeries for VSE/ESA
configuration 549
multi-hopping 14
passing messages through system 41
passing through intermediate queue managers 14
putting messages on remote queues 38
QM-concentrators 31
queue name resolution 54
receiving messages 40
renaming a channel 115
reply-to-queue 47, 48
reply-to-queue alias 28
sending messages 5, 17
sending messages in both directions 5
separating message flows 42
setting up communication for OS/2 and Windows systems
defining a NetBIOS connection 135
defining a TCP connection 130
defining an LU 6.2 connection 132
defining an SPX connection 138
setting up communication in Compaq NonStop Kernel
SNA channels 307
TCP channels 309
setting up communication in Compaq OpenVMS Alpha
defining a DECnet Phase IV connection 304
defining a DECnet Phase V connection 304
defining an LU 6.2 connection 299
setting up communication in UNIX systems
defining a TCP connection 191
defining an LU 6.2 connection 194
setting up communication in WebSphere MQ for z/OS
LU 6.2 connection 364, 446
TCP connection 361
setting up communications in Compaq OpenVMS Alpha
defining a TCP connection 296
sharing a transmission queue 14
SINIX and DC/OSx configuration files 329
examples (continued)
starting a channel 115, 490
triggering 21, 122
using multiple channels 15
using the remote queue definition object 37
WebSphere MQ for AIX configuration 197
WebSphere MQ for HP-UX configuration 219
WebSphere MQ for iSeries configuration 521
WebSphere MQ for Solaris configuration 257
WebSphere MQ for Windows configuration 167
WebSphere MQ for z/OS configuration 365, 421, 447
exit 399
exit wait descriptor structure 685
ExitBufferAddr parameter 616
ExitBufferLength parameter 616
ExitData field 674
ExitDataLength field 645
ExitId field 668
ExitNameLength field 644
ExitNumber field 676
ExitParms parameter 622
EXITPATH stanza of qm.ini file 715
ExitReason field
MQCXP structure 668
MQTXP structure 682
ExitResponse field
MQCXP structure 670
MQTXP structure 683
ExitResponse2 field 671
ExitSpace field 676
ExitUserArea field
MQCXP structure 673
MQTXP structure 682
Feedback
MQCXP structure 673
MQTXP structure 684
GroupId 683
HeaderLength 675
HeartbeatInterval 642
KeepAliveInterval 651
LocalAddress 651
LongMCAUserIdLength 648
LongMCAUserIdPtr 649
LongRemoteUserIdLength 649
LongRemoteUserIdPtr 649
LongRetryCount 633
LongRetryInterval 633
MaxMsgLength 635
MaxSegmentLength 673
MCAName 632
MCASecurityId 649
MCAType 638
MCAUserIdentifier 637
ModeName 632
MsgExit 634
MsgExitPtr 645
MsgExitsDefined 645
MsgRetryCount
MQCD structure 641
MQCXP structure 674
MsgRetryExit 640
MsgRetryInterval
MQCXP structure 642
MQTXP structure 684
fields
BatchHeartbeat 652
BatchInterval 643
BatchSize 632
CapabilityFlags 675
ChannelName 629
ChannelType 630
ClusterPtr 648
ClustersDefined 648
ConnectionName 639
DataConversion 636
DataId 683
DataLength 683
Desc 631
details of receiver channel panel 409
Index 729
Getting started with intra-group queuing 465
GroupId field 683

H
HBINT attribute 87
Hconn parameter 624
HeaderLength field 675
heartbeat interval 87
HeartbeatInterval field 642
help
pull-down menus 404, 405
z/OS using CICS 404
help menu-bar choice 404, 405
how to use
channel planning form 699

IBM Communications Server for Windows NT 172
ICE communications example 317
in doubt 82
in doubt channels, manual
resynchronization 70
in doubt message on channel, resolve on
z/OS 352
in doubt messages, commit or back out
Compaq NonStop Kernel 119
Compaq OpenVMS Alpha 119
OS/2 119
OS/400 501
UNIX systems 119
Windows systems 119
INACTIVE channel state 62, 65
inbound channels
with shared queuing 441
Inbound channels with shared queuing 441
ini file 74
initial data negotiation 20, 61
initialization data set, OS/390 without
CICS 73
initialization file 73
initiator for channel
AIX, OS/2, HP-UX, Solaris, and
Windows systems 122, 123
z/OS 346
installing
CICS communication connection 419
integrity of delivery 22
intercommunication
concepts 3, 16, 22
example 549
example configuration 101
intercommunication example 549, 567
intercommunication examples
MQSeries for AT&T GIS UNIX 243
MQSeries for Compaq Tru64
UNIX 215
MQSeries for OS/2 Warp 143
MQSeries for VSE/ESA 549
WebSphere MQ for AIX 197
WebSphere MQ for HP-UX 219
WebSphere MQ for iSeries 521

intercommunication examples (continued)
WebSphere MQ for Solaris 257
WebSphere MQ for Windows 167
WebSphere MQ for z/OS 365, 421,
447
interface
generic 439
interfaces
IUCV 360
Sterling Software SOLVE:TCPaccess
interface 360
intersystem communication (ISC) 417
intra-group queuing
agent 463
benefits 463
concepts of 461
configurations 466
example 475
getting started 465
intra-group queuing agent 461
limitations 464
messages 470
security 472
shared transmission queue 463
specific properties 472
termology 462
with clustering 468
with distributed queuing 466
intra-group queuing agent
intra-group queuing 461
Intra-group queuing agent 463
Intra-group queuing and the intra-group
queuing agent 461
Intra-group queuing benefits 463
Intra-group queuing concepts 461
Intra-group queuing configurations 466
Intra-group queuing example 475
Intra-group queuing limitations 464
Intra-group queuing security 472
Intra-group queuing specific
properties 472
Intra-group queuing with clustering 468
Intra-group queuing with distributed
queuing 466
ISC (intersystem communication) 417
IUCV interface 360

journaling 581

K
KAINT attribute 87
KEEPALIVE 66
OS/2 139, 159
OS/400 512
UNIX systems 194
KeepAlive interval 87
KeepAliveInterval field 651
keyboard functions
function keys
z/OS using CICS 385
z/OS using CICS
clear key 386

G
generic interface 439
Generic interface 439
getting started
intra-group queuing 465

730  WebSphere MQ Intercommunication
loopback testing 56

LU 6.2
mode name 89
responder processes 309
settings
OS/2 132
OS/400 512
UNIX systems 194
Windows systems 132
TP name 89
LU 6.2 connection
MQSeries for AT&T GIS UNIX 243
MQSeries for Compaq NonStop
Kernel 307
MQSeries for OS/2 Warp 143
setting up
OS/2 129
OS/400 51
UNIX systems 191
Windows NT 129
z/OS 364, 446
z/OS using CICS 418
WebSphere MQ for AIX 197
WebSphere MQ for Compaq
OpenVMS Alpha 295
WebSphere MQ for HP-UX 219
WebSphere MQ for iSeries 521
WebSphere MQ for Solaris 258, 272
WebSphere MQ for VSE/ESA 549
WebSphere MQ for Windows 167
WebSphere MQ for z/OS 365, 421, 447
WebSphere MQ for z/OS with
CICS 417, 424
LU62
stanza of qm.ini file 715

M
maximum
active channels 64
current channels 64
message length 90
transmission size 90
MAXMSGL attribute 90
MaxMsgLength field 635
MaxSegmentLength field 673
MCA
adopting 67, 691
caller 10
name 91
responder 10
type 91
user 91
user-written 75
MCANAME attribute 91
MCAName field 632
MCASecurityId field 649
MCATYPE attribute 91
MCAType field 638
MCAUSER attribute 91
MCAUserIdentifier field 637
message
committed
Compaq NonStop Kernel 119
Compaq OpenVMS Alpha 119
OS/2 119
message (continued)
committed (continued)
OS/400 501
UNIX systems 119
Windows systems 119
concentrating 44
converting 85
diverting flows 45
equency 571
error 687
for distribution list 26
for distribution list 46
passing through system 41
putting on remote queue 37
queue name translations 53
receiving 40
return routing 53
return to sender 72
routing 39
sending and receiving 58
separating flows 42
sequence numbering 54
sequential retrieval 55
splitting 59
undeliverable 71
message channel
caller 10
cluster-receiver 8, 10
cluster-sender 8, 9
receiver 8
requester 8
requester-server 8
sender 8
sender-receiver 8
server 8
server-receiver 9
message channel agent
example
Compaq NonStop Kernel 326
Compaq OpenVMS Alpha 326
OS/2 326
OS/400 544
UNIX systems 326
Windows NT 326
z/OS 380
Local queue definition
Local queue manager 3
LOCALADDR attribute 88
LocalAddress field 651
location name 41
log
effect 118, 694
file, @SYSTEM 694
logs for errors 118
long retry count attribute 88
long retry interval attribute 89
LongMCAUserIdLength field 648
LongMCAUserPtr field 649
LongRemoteUserIdLength field 649
LongRemoteUserPtr field 649
LongRetryCount field 633
LongRTY attribute 88
LONGTMR attribute 89
message
assured delivery 22
messages (continued)
back out in-doubt messages
OS/400 501
commit in-doubt messages
OS/400 501
intra-group queuing 470
resolve in-doubt messages
OS/400 501
sending 17
Messages
back out in-doubt messages 119
commit in-doubt messages 119
resolve in-doubt messages 119
messages and codes 71
Messages put to
SYSTEM.QSG.TRANSMIT
.QUEUE 470
mode name 89
MODENAME attribute 89
ModeName field 632
monitoring and controlling channels
Compaq NonStop Kernel 109
Compaq OpenVMS Alpha
systems 109
OS/2 109
OS/400 485
UNIX systems 109
Windows 109
z/OS 341
z/OS using CICS 383
monitoring channels 60
moving service component 4
MQ_* values 703
MQ_CHANNEL_AUTO_DEF_EXIT
call 619
MQ_CHANNEL_EXIT call 614
MQ_TRANSPORT_EXIT call 622
MQCD structure 626
MQCD, channel definition structure 585
MQCXP structure 666
ExitReason field 581
ExitSpace field 581
MQCXP_* values 666
MQCXP, channel exit parameter
structure 585
MQFB_* values 673
MQI channels 8
MQIBindType 126
MQRMH, reference-message header 582
mqs.ini 74
MQSeries for AT&T GIS UNIX
channel configuration 252
channel-exit programs 598
configuration 252
intercommunication example 243,
257
LU 6.2 connection 243
TCP connection 251
MQSeries for DIGITAL UNIX (Compaq
Tru64 UNIX)
channel configuration 216
configuration 215
intercommunication example 215
TCP connection 215
MQSeries for OS/2 Warp
channel configuration 160
channel-exit programs 589

732

WebSphere MQ Intercommunication

MQSeries for OS/2 Warp (continued)
configuration 159
intercommunication example 143,
165
LU 6.2 connection 143
NetBIOS connection 157
SPX connection 157
TCP connection 155
MQSeries for SINIX and DC/OSx
channel-exit programs 600
MQSeries for VSE/ESA
channel configuration 554
configuration 554
configuration worksheet 549
LU 6.2 connection 549
MQSINI 74
MQTXP structure 681
MQTXP_* values 681
MQXCC_* values
MQCXP structure 670
MQTXP structure 683
MQXCP_VERSION_5, of MQCXP
structure 581
MQXQH, transmission header 582, 583
MQXR_* values
MQCXP structure 668
MQTXP structure 682
MQXR_INIT, ExitReason value 581
MQXR_XMIT, ExitReason value 581
MQXR2_* values 671
MQXT_* values 668
MQXUA_* values
MQCXP structure 682
MQTXP structure 673
MQXWAIT call 624
MQXWD structure 685
MQXWD_* values 685
MRDATA attribute 92
MREXIT attribute 92
MRO (multiregion operation) 417
MRRTY attribute 92
MRTMR attribute 93
MSGDATA attribute 92
MSGEXIT attribute 92
MsgExit field 634
MsgExitPtr field 645
MsgExitsDefined field 645
MsgRetryCount field
MQCD structure 641
MQCXP structure 674
MsgRetryExit field 640
MsgRetryInterval field
MQCD structure 642
MQCXP structure 674
MsgRetryReason field 674
MsgRetryUserData field 641
MsgUserData field 636
MsgUserDataPtr field 646
multi-hopping 14
multiple message channels per
transmission queue
Compaq NonStop Kernel 61
Compaq OpenVMS Alpha 61
OS/2 61
UNIX systems 61
Windows systems 61
z/OS using CICS 414

multiple queue managers 134
multiregion operation (MRO) 417

N
name resolution
conflicts 53
convention 53
description 711
introduction 26
location 41
queue name translations 53
restriction 48
NDF file configuration 134
negotiations on startup 61, 689
NetBIOS 4, 135
NETBIOS
stanza of qm.ini file 715
NetBIOS connection
MQSeries for OS/2 Warp 157
OS/2 129
WebSphere MQ for Windows 180
Windows NT 129
NetBIOS products, in example
configurations 102
NetBIOS, example configurations 102
network infrastructure, example
configurations 102
network planner 31
network-connection priority 93
networking 41
networking considerations 52
NetworkPriority field 648
networks 29
node centric 36
nonpersistent message speed 93
NonPersistentMsgSpeed field 643

O
objects
creating
Compaq NonStop Kernel 112
Compaq OpenVMS Alpha 112
default 113
OS/2 112
OS/400 488
UNIX systems 112
Windows systems 112
defining 413
z/OS 357
security 124, 509
operator commands
OS/400 486
options
alter 402
change 498
copy 399, 498
create 400, 497
display 499
display settings 397
display status 396, 499
end 500
exit 399, 405
find 402
ping 398, 499


options (continued)
reset 394, 501
resolve 119, 395
OS/400 501
resync 393
save 405
start 389, 499
stop
z/OS using CICS 391
OS/400
KEEPALIVE 512
outbound channels
with shared queuing 441
Outbound channels with shared queuing 441

panels (continued)
Work with channel status
OS/400 494
work-with-channel choices
OS/400 496
z/OS using CICS, message channel list 386
parameters
AgentBuffer 615
AgentBufferLength 615
ChannelDefinition
MQ_CHANNEL_AUTO_DEF_EXIT
call 619
MQ_CHANNEL_EXIT call 614
ChannelExitParms
MQ_CHANNEL_AUTO_DEF_EXIT
call 619
MQ_CHANNEL_EXIT call 614
CompCode 624
DataLength 614
DestAddress 622
DestAddressLength 622
ExitBufferAddr 616
ExitBufferLength 616
ExitParms 622
Hconn 624
Reason 624
WaitDesc 624
parameters, receiving 59
PartnerName field 675
password 93
Password field 637
PAUSED channel state 62, 66
peer recovery
with queue-sharing 442
Peer recovery 442
peer-shared-channel retry on z/OS about 692
performance
channel 189
ping 398, 499
Compaq NonStop Kernel 116
Compaq OpenVMS Alpha 116
OS/2 116
problem determination 687
UNIX systems 116
Windows systems 116
ping channel
z/OS 351
ping with LU 6.2 116, 499
pipelining
in MCA message transfer 189
parameter in qm.ini file 189
planning
message channel planning example
z/OS using CICS 431
planning form 699
port (continued)
WebSphere MQ for Windows systems 130
WebSphere MQ for z/OS 346, 371, 426, 454
preparation
getting started
Compaq NonStop Kernel 112
Compaq OpenVMS Alpha 112
OS/2 112
OS/400 488
UNIX systems 112
Windows systems 112
preparing channels 60
preparing WebSphere MQ for iSeries 503
problem determination 687
channel refuses to run 689
channel startup negotiation errors 689
channel switching 693
clients 694
connection switching 694
conversion failure 691
data structures 692
dead-letter queue 688
error messages 687
retrying the link 692
scenarios 687
transmission queue overflow 688
triggered channels 690
undelivered-message queue 688
user-exit programs 692
using the PING command 687
validation checks 688
process definition example
Compaq NonStop Kernel 326
Compaq OpenVMS Alpha 326
OS/2 326
UNIX systems 326
Windows NT 326
process definition example
Compaq NonStop Kernel 325
Compaq OpenVMS Alpha 325
OS/2 325
UNIX systems 325
Windows NT 325
process definition for triggering
Compaq NonStop Kernel 121
Compaq OpenVMS Alpha 121
OS/2 121
OS/400 505
UNIX systems 121
Windows systems 121
z/OS 357
z/OS using CICS 390, 413
process security 94
processing problems 71
profile name, CICS 83
programs
AMQCRCTA 507
AMQCRS6A 123
AMQCRSTA 123, 507
AMQRMCLA 507
RUNMQCHI 123
RUNMQCHL 123
programs (continued)
RUNMQLSR 123
pull-down menus
channel 405
edit 399
help (channel definition panels) 405
help (message channel list panel) 404
selected 389
view 403
PUT authority 94
PUTAUT attribute 94
PutAuthority field 635
putting messages 37
on remote queues 37
to distribution lists 46

Q
qm.ini 74
Channels stanza 189
stanzas used for distributed queuing 715
QMName field 631
QMName attribute 95
queue
destination 42
reply-to 47
queue manager
alias 25, 36
receiving 40
interconnection procedure example 431
multiple 131
name 95
alias 42
types 3
queue manager alias 25, 36
introduction 26
receiving 40
queue name
resolution 711
how it works 714
what is it? 713
translations 53
queue sharing
peer recovery with 442
queue-sharing clusters and queues
create a transmission queue 121, 503
defining 413
z/OS 357
queuing
shared 439
quiescing channels 67

R
RCVDATA attribute 95
RCVEXIT attribute 95
reason codes
numeric list 706
Reason parameter 624
receive exit 12
name 95
program 578
user data 95
ReceiveExit field 635
ReceiveExitPtr field 647
ReceiveExitsDefined field 645
receiver
channel 8
channel definition example
  Compaq NonStop Kernel 325
  Compaq OpenVMS Alpha 325
  OS/2 325
  OS/400 544
  UNIX systems 325
  Windows NT 325
  z/OS 379
channel panel
details 409
panel settings
OS/390 using CICS 409
receiver channel definition example
  Compaq NonStop Kernel 327
  Compaq OpenVMS Alpha 327
  OS/2 327
  OS/400 545
  UNIX systems 327
  Windows NT 327
  z/OS 380
overview 5
ReceiveUserData field 637
ReceiveUserDataPtr field 647
receiving
messages 40, 58
on DECnet Phase IV 304
on LU 6.2
  Compaq NonStop Kernel 309
  OS/2 134
  UNIX systems 195
  Windows systems 134
  z/OS 364, 446
on SPX
  OS/2 139, 159
  Windows NT 182
  Windows systems 139
on TCP
  Compaq NonStop Kernel 321
  Compaq OpenVMS Alpha 296
  OS/2 130
  OS/400 512
  UNIX systems 192
  Windows systems 130
  z/OS 362, 445
receiving messages 40, 58
receiving on DECnet Phase IV 304
receiving on LU 6.2
  Compaq NonStop Kernel 309
  OS/2 134
  UNIX systems 195
  Windows systems 134
  z/OS 364, 446
receiving on SPX
  OS/2 139, 159
  Windows NT 182
  Windows systems 139
receiving on TCP
  Compaq NonStop Kernel 321
  Compaq OpenVMS Alpha 296
  OS/2 130
  OS/400 512
  UNIX systems 192
  Windows systems 130
  z/OS 362, 445
recovery with queue-sharing peer 442
reference-message header
message exit program 582
Registration time for DDNS 691
registry 73, 74, 132, 135, 141, 181, 592
remote queue definition example
  Compaq NonStop Kernel 325
  Compaq OpenVMS Alpha 325
  OS/2 325
  OS/400 542
  UNIX systems 325
  Windows NT 325
  z/OS 378
introduction 15, 25
what it is 14
remote queue manager 3
RemotePassword field 640
RemoteSecurityId field 650
RemoteUserIdentifier field 639
renaming a channel
  Compaq NonStop Kernel 115
  Compaq OpenVMS Alpha 115
  OS/2 115
  OS/400 494
  UNIX systems 115
  Windows systems 115
  z/OS using CICS 389
reply-to alias 36
reply-to queue 47
alias definition 28
alias example 48
aliases 25
preparing for 29
specifying 28
reply-to queue definition example
  Compaq NonStop Kernel 325
  Compaq OpenVMS Alpha 325
  OS/2 325
  OS/400 544
  UNIX systems 325
  Windows NT 325
  z/OS 379
requester channel 8
requester channel settings panel, details 411
REQUESTING channel state 62
Reserved field 682
Reserved1 field 685
Reserved2 field 685
Reserved3 field 686
reset 119, 394, 501
RESET CHANNEL command 690
reset channel sequence numbers, z/OS 351
 resolve 395
RESOLVE CHANNEL command 689
resolve in-doubt message on channel, z/OS 352
resolve in-doubt messages 119
OS/400 301
resolve option 119
OS/400 301
responder LU6.2 80, 117, 309
MCA 10
responder MCA 10
responder process 80, 117, 309
restarting channels 61
restarting stopped channels 69
rerec 393
RETRY channel state 62, 65
retry considerations 692
RetryCount field 683
retrying the link, problem determination 692
return routing 53
return to sender 72
reusing exit programs 600
routing entry add 518
class 519
routing entry class 519
routing messages 39
run channel 115, 490
run channel initiator 122
runmqch command AIX, OS/2, HP-UX, Solaris, and Windows systems 122, 123
RUNMQCHI command 123
RUNMQCHL command 123
RUNMQCHL command 123
S
sample security exit 610
save option 405
scenarios, problem determination 687
SCF configuration file, example 309
SCYDATA attribute 96
SCYEXIT attribute 96
security context 94
exit 12
exit name 96
exit program 574
exit program, overview 572
exit user data 96
intra-group queuing 472
levels for exit programs 126
message channel agent 94
objects
Compaq NonStop Kernel 124
Compaq OpenVMS Alpha 124
OS/2 124
UNIX systems 124
WebSphere MQ for iSeries 509
Windows systems 124
process 94
security exit
sample 610
Security Services Programming Interface (SSPI) 610
SecurityExit field 634
SecurityUserData field 636
segregating messages 15
selected menu-bar choice 389
selecting a channel 492
z/OS using CICS 386
send exit 12
exit name 96
exit program 578
message 57
send exit multiple send exits 581
send exit user data 96
SEND DATA attribute 96
sender channel 8
sender channel definition example
Compaq NonStop Kernel 325, 327
Compaq OpenVMS Alpha 325,
327
OS/2 325, 327
OS/400 543, 545
UNIX systems 325, 327
Windows NT 325, 327
z/OS 379, 380
overview 5
sender channel settings details 408
SEND EXIT attribute 96
SendExit field 634
SendExitPtr field 646
SendExitsDefined field 645
sending messages 17, 58
on DECNiel Phase IV 304
on SPX
OS/2 138
Windows NT 138
on TCP
Compaq OpenVMS Alpha 296
OS/2 130
Windows NT 130
sending on TCP 191
SendUserData field 636
SendUserDataPtr field 646
SeqNumberWrap field 635
sequence number wrap 96
sequence numbering 54
sequence numbers 59
reset, z/OS 351
sequential delivery 97
sequential retrieval of messages 55
SEQWRAP attribute 96
server AT&T G1S SNA 246
channel 8
channel settings panel, detail 410
server-connection channel 8
service class of 439
service, class of 47
SessionId field 683
setting up
CICS communication for z/OS 417
communication
Compaq NonStop Kernel 307
setting up (continued)
communication (continued)
Compaq OpenVMS Alpha systems 295
OS/2 129
OS/400 511
UNIX systems 191
Windows NT 129
shared queuing 439
shared queuing benefits of 441
components of 439
concepts of 439
inbound channels 441
message channel agents with 441
outbound channels 441
synchronization queue 441
transmission 440
triggering with 440
shared transmission queue
intra-group queuing 463
Shared transmission queue for intra-group queuing 463
sharing channels 14
short retry count 97
interval 97
ShortConnectionName field 631
ShortRetryCount field 633
ShortRetryInterval field 633
SHORTRTY attribute 97
SHORTTMR attribute 97
side object
OS/400 514
SINIX and DC/OSx configuration files 329
SNA 4 configuration, Compaq OpenVMS Alpha 299
listener process, ending 302
products, in example configurations 102
Server 133
SNAP-IX
configuration parameters 272
establishing a session 276
explanation of terms 274
operation 288
sender-channel definitions 292
SNAPplus2 223
SNAX communications examples 309
SO_KEEPALIVE
Compaq OpenVMS Alpha 297
OS/2 132
OS/400 512
UNIX systems 194, 297
Windows systems 132
source queue manager 3
specific properties
intra-group queuing 472
splitting messages 59
SPX 4
connection
MQSeries for OS/2 Warp 157
OS/2 129
WebSphere MQ for Windows 181
Index 735
example configurations 102
KEEPALIVE, OS/2 139, 159
products, in example
configurations 102
stanz of qm.ini file 715
Windows NT 129
SSLCAUTH attribute 98
SSLCIPH attribute 98
SSLcipherspec field 650
SSLClientAuth field 651
SSLPeerNameLength field 650
SSLPeerNamePtr field 650
SSPI (Security Services Programming
Interface) 610
stanza file 74
stanza, in qm.ini file
configurations 102
channels 189
channel 61
Comp qNKS Kernel 115
Comp qVMS Alpha 115
OS/2 115
UNIX systems 115
z/OS 349
channel initiator, z/OS 346
channel listener, z/OS 348
DQM panels, z/OS using CICS 384
option 389, 499
STARTING channel state 62
startup dialog 572
state, channel 61
status
display channel 114
work with channel 494
status panel 396, 499
status, channel 59
Sterling Software SOLVE:TCPaccess
interface 360
stop
channel 67, 118, 391
channel initiator, z/OS 347
channel listener, OS/390 349
channel, z/OS 352
channel, z/OS using CICS 415
cntrolled 501
immediate 501
option 391
queue 119, 393
stop channel initiator 123
stop force 119
stop immediate 392
STOPPED channel state 62, 65
stopped channels, restarting 69
STOPPING channel state 62
Strmqm command 113
Strucl field
MQCX struct 666
MQTXP 681
MQXWD 685
StruclLength field 644
structure
MQCX 581
structures
MQCD 626
structures (continued)
MQCX 666
MQTXP 681
MQXWD 685
SunLink Version 9.1 262
switching channels 693
synchronization
with shared queuing 441
Synchronization queue with shared
queuing 441
synchronization queue, z/OS 358
synchronizing channels 393, 572
syncpoint introduction 81
system extension 126, 510
system extensions
user-exit programs
Comp qNKS Kernel 126
Comp qVMS Alpha 126
OS/2 126
UNIX systems 126
WebSphere MQ for iSeries 510
Windows systems 126
system identifier, target 99
SYSTEM.CHANNEL.INITQ queue
Comp qVMS Alpha 323
OS/2 323
OS/400 541
UNIX systems 323
Windows NT 323
z/OS 341, 358
SYSTEM.CHANNEL.REPLY.INFO
queue 341, 358
SYSTEM.CHANNEL.SYNCQ 358
TCP target queue manager 3
target system identifier 99
TCP channels, Comp qNKS Kernel 309
connection
listener backlog 131, 139, 193,
362, 512
MQSeries for AT&T GIS
UNIX 251
MQSeries for OS/2 Warp 155
WebSphere MQ for AIX 208
WebSphere MQ for Comp qNKS Kernel 307
WebSphere MQ for Comp qVMS Alpha 295
WebSphere MQ for Comp qTru64
UNIX 215
WebSphere MQ for HP-UX 237
WebSphere MQ for iSeries 532
WebSphere MQ for Solaris 288
WebSphere MQ for VSE/ESA 554
WebSphere MQ for z/OS 371, 453
Windows 180
example configurations 102
listener backlog option 131, 139, 193,
362, 512
OpenEdition MVS sockets 360
products, in example
configurations 102
stanz of qm.ini file 715
TCP connection (continued)
stanz of QMINI file 715
TCP connection
setting up
OS/2 129
UNIX systems 191
Windows NT 129
z/OS 361
TCP KEEPALIVE
OS/2 132
OS/400 512
UNIX systems 194
Windows systems 132
TCP/IP 4
TCP/IP communications example 321
TCP/IP KEEPALIVE 66
Comp qVMS Alpha 297
UNIX systems 297
TCPware 298
terminology
intra-group queuing 462
test channel, z/OS 351
testing connections, lookback testing 56
threads
multiple 189
time-out 86
TPNAME and TPATH
OS/2 132
OS/400 513
UNIX systems 194
Windows systems 132
TPNAME attribute 89
TpName field 632
transaction
identifier, CICS 99
program name 89
transactions
CEDA 418
CKMC 384
CKSG 413
transmission
shared queuing 440
transmission header
alias
definition 26
message exit program 582
message-retry exit program 583
queue name 36
transmission of messages
maximum transmission size 581
transmission buffer 581
transmission protocol 99
transmission queue
definition of 10
eample definition
Comp qNKS Kernel 325
Comp qVMS Alpha 325
OS/2 325
OS/400 543
UNIX systems 325
Windows NT 325
z/OS 378
multiple message channels
Comp qNKS Kernel 61
Comp qVMS Alpha 61
OS/2 61
UNIX systems 61
transmission queue (continued) multiple message channels (continued)  
Windows systems 61  
z/OS using CICS 414  
overflow 688  
selecting 42  
sharing 14  
Transmission queue 440  
transmission queue definition example  
Compaq NonStop Kernel 326  
Compaq OpenVMS Alpha 326  
OS/2 326  
OS/400 545  
UNIX systems 326  
Windows NT 326  
z/OS 380  
transmission queue name 99  
transport type 99  
supported 4  
transport-retry exit  
introduction 12  
transport-retry exit program 584  
TransportType field  
MQCD structure 630  
MQTP structure 682  
triggered channels, problem determination 690  
triggering  
channels 20  
Compaq NonStop Kernel 121  
Compaq OpenVMS Alpha 121  
OS/2 121  
UNIX systems 121  
WebSphere MQ for iSeries 505  
Windows systems 121  
z/OS 357  
z/OS using CICS 390  
MCAs  
z/OS using CICS 413  
with shared queuing 440  
Triggering with shared queuing 440  
TRPTYPE attribute 99  
thrust applications 12, 126  
type, bind 126  
types of channel 83  

U  
UCD products, in example configurations 102  
UDP  
example configurations 102  
UDP connection  
setting up  
UNIX systems 191  
WebSphere MQ for AIX 209  
UDP transport-retry exit, configuring 584  
undeliverable message 71  
UNIX  
KEEPALIVE 194  
user ID 100, 125  
user identifier service 125  
user-exit  
programs 692  
user-exit programs 571, 610  
user-exit programs (continued)  
problem determination 692  
security levels 126  
system extension  
Compaq NonStop Kernel 126  
Compaq OpenVMS Alpha 126  
OS/2 126  
OS/400 510  
UNIX systems 126  
Windows systems 126  
writing and compiling 585  
user-written MCAs 75  
USERDATA parameter 390, 413, 505  
z/OS 357  
USERID attribute 100  
UserIdentifier field 637  

V  
validation  
checks 688  
command 71  
of user IDs 581  
panel data 71  
values supplied by OS/390 using CICS 401, 404  
values supplied by WebSphere MQ for iSeries 497  
Version field  
MQCD structure 629  
MQCXP structure 667  
MQTP structure 681  
MQXWD structure 685  
view  
activities  
OS/390 using CICS 403  
menu-bar choice 403  
VSAM 383  

W  
WaitDesc parameter 624  
WAITING channel state 62  
WebSphere MQ for AIX  
channel configuration 210  
channel-exit programs 593  
configuration 209  
intercommunication example 197, 214  
LU 6.2 connection 197  
TCP connection 208  
UDP connection 209  
WebSphere MQ for Digital OpenVMS channel-exit programs 595  
problem solving 303  
setting up communication 295  
SNA configuration 299  
WebSphere MQ for HP-UX  
channel configuration 238  
channel-exit programs 597  
configuration 238  
intercommunication example 219, 243  
LU 6.2 connection 219  
TCP connection 237  
WebSphere MQ for iSeries  
channel configuration 535  
channel-exit programs 589  
configuration 534  
intercommunication example 521, 540  
LU 6.2 connection 521  
TCP connection 532  
WebSphere MQ for Solaris  
channel configuration 289  
channel-exit programs 599  
configuration 289  
intercommunication example 257, 295  
LU 6.2 connection 258  
TCP connection 288  
WebSphere MQ for Tandem NonStop Kernel  
channel-exit programs 600  
setting up communication 307  
WebSphere MQ for Windows  
channel configuration 184  
channel-exit programs 591  
configuration 183  
intercommunication example 167, 189  
LU 6.2 connection 167  
NetBIOS connection 180  
SPX connection 181  
TCP connection 180  
WebSphere MQ for z/OS  
channel configuration 372, 454  
configuration 371, 454  
intercommunication example 365  
LU 6.2 connection 365, 421, 447  
reset channel sequence numbers 351  
resolving in-doubt message on channel 352  
TCP connection 371, 453  
WebSphere MQ for z/OS using CICS channel-exit programs 588  
WebSphere MQ for z/OS without CICS channel-exit programs 587  
wide-band links 31  
Windows 3.1 client, channel-exit programs 591  
Windows 95 and Windows 98 client, channel-exit programs 591  
work with channel status 494  
work with status 499  
work-with-channel choices 496  
workload  
balanced 442  
Workload-balanced channel start 442  
worksheet  
MQSeries for AT&T GIS UNIX configuration 243  
MQSeries for OS/2 Warp configuration 143  
WebSphere MQ for AIX configuration 197  
WebSphere MQ for HP-UX configuration 219  
WebSphere MQ for iSeries configuration 521  
WebSphere MQ for Solaris configuration 258, 272  
Index 737
worksheet (continued)
WebSphere MQ for Windows
configuration 168
WebSphere MQ for z/OS
configuration 366, 422, 448
writing your own message channel
agents 75
WRKCLS command 519
WRKSBSD command 517

X
XMITQ attribute 99
XmitQName field 631

Z
z/OS connections
connecting systems 417
LU 6.2 417
Sending your comments to IBM

If you especially like or dislike anything about this book, please use one of the methods listed below to send your comments to IBM.

Feel free to comment on what you regard as specific errors or omissions, and on the accuracy, organization, subject matter, or completeness of this book.

Please limit your comments to the information in this book and the way in which the information is presented.

To make comments about the functions of IBM products or systems, talk to your IBM representative or to your IBM authorized remarketer.

When you send comments to IBM, you grant IBM a nonexclusive right to use or distribute your comments in any way it believes appropriate, without incurring any obligation to you.

You can send your comments to IBM in any of the following ways:

• By mail, to this address:
  User Technologies Department (MP095)
  IBM United Kingdom Laboratories
  Hursley Park
  WINCHESTER,
  Hampshire
  SO21 2JN
  United Kingdom

• By fax:
  – From outside the U.K., after your international access code use 44–1962–816151
  – From within the U.K., use 01962–816151

• Electronically, use the appropriate network ID:
  – IBM Mail Exchange: GBIBM2Q9 at IBMMAIL
  – IBMLink™: HURSLEY(IDRCF)
  – Internet: idrcf@hursley.ibm.com

Whichever method you use, ensure that you include:

• The publication title and order number
• The topic to which your comment applies
• Your name and address/telephone number/fax number/network ID.
Spine information:

IBM    WebSphere MQ    WebSphere MQ Intercommunication